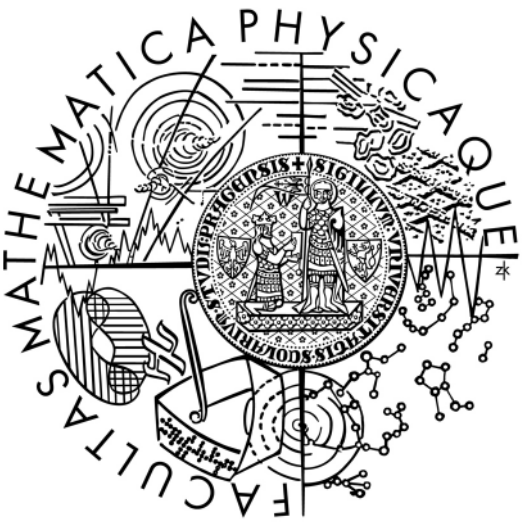
Univerzita Karlova v Prahe

Matematicko-fyzikálna fakulta

BAKALÁRSKÁ PRÁCA



Michal Rada

Aplikácia na správu nehnuteľností

Katedra softwarového inžinierstva

Vedúci bakalárskej práce: RNDr. Jan Kofroň, Ph.D.

Študijní program: Správa počítačových systémov

2008/2009

Poďakovanie

Týmto by som chcel poďakovať vedúcemu mojej bakalárskej práce, RNDr. Janu Kofroňovi, Ph. D. za pomoc a všetok čas, ktorý mi venoval.

Prehlasujem, že som svoju bakalársku prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím so zapožičaním práce.

V Prahe

dňa Michal Rada

12.12.2008

Názov práce: Aplikácia na správu nehnuteľností

Autor: Michal Rada

Katedra (ústav): Katedra softwarového inžinierstva

Vedúci bakalárskej práce: RNDr. Jan Kofroň, Ph. D.

e-mail vedúceho: Jan.Kofron@mff.cuni.cz

Abstrakt: Cieľom bakalárskej práce je navrhnúť a implementovať Internetovú aplikáciu, ktorá má za úlohu zjednodušiť prácu majiteľom nehnuteľností so správou nehnuteľností, nájomníkov, výberom poplatkov a nájomného a sledovaním a výpočtom návratnosti investícii do nehnuteľností. Aplikácia by mala obsahovať nielen rozhranie pre prenajímateľa, ale aj rozhranie pre nájomníkov, ktoré by poskytovalo odpovedajúce informácie druhej strane a umožňovalo nájomníkom kontaktovať svojho prenajímateľa. Aplikácia by mala byť navrhnutá s ohľadom na ľahkú použiteľnosť a jednoduchosť ovládania, ideálne nezávisle na platforme.

Kľúčové slová: správa, nehnuteľnosť, nájomník, financie

Title: Application for Real Estate Management

Author: Michal Rada

Department: Department of Software Engineering

Supervisor: RNDr. Jan Kofroň, Ph. D.

Supervisor's e-mail address: Jan.Kofron@mff.cuni.cz

Abstract: The goal of the bachelor thesis is to design and implement an Internet application which aims at simplification of the work of the real estate owners regarding the management of real estates, renters, fees, and rent, monitoring, and calculation of return of investments. The application should contain not only the interface for the owner, but also an interface for renters, which would offer appropriate information to the other side and would allow communication between renters and the owners. The application should be designed with respect to easy use and simple control, in ideal case in a platform independent way.

Keywords: management, property, tenant, finances

Úvod

Aplikácia na správu nehnuteľností slúži ich majiteľom na evidenciu jednotlivých nehnuteľností a informácií spojených s touto evidenciou. V aplikácii majú jednotliví majitelia možnosť evidovať nájomníkov, ktorým prenajali danú nehnuteľnosť, služby, ktoré sú spojené s danou nehnuteľnosťou (topenie, elektrina, odvoz odpadu, a iné) a správu financií. V rámci správy financií aplikácia umožňuje vlastníkom nehnuteľností rýchly prístup k informáciám o nájme, platbách, stave nehnuteľností a kontaktovať a informovať prenajímateľa prostredníctvom daného rozhrania. Zároveň aplikácia automaticky generuje predpisy k platbám za nájom a iné služby, ktoré si majiteľ v aplikácii zadefinuje.

Na tému spravovania nehnuteľností je v dnešnej dobre vytvorených niekoľko programov, ktoré komplexne spracovávajú danú problematiku. Sú to programy, ktoré používajú bytové družstvá, rôzne správcovské a developerské firmy, ktoré v tejto oblasti pracujú a orientujú sa v nej. Väčšinou si každá z týchto firiem vytvára vlastný software vzhľadom k ich potrebám. Jedná sa o spoplatňované programy, ktoré pracujú rádovo so stovkami nehnuteľností. Každý z týchto programov je adresovaný subjektom podobného zamerania, ako sú jednotlivé firmy. Teda sú to buď ďalšie firmy, prípadne jednotlivci, ktorí vlastnia a prenajímajú väčší počet nehnuteľnosti.

Pri hľadaní na Internete sa mi podarilo nájsť niekoľko programov (www.jirra.cz, www.des.cz, www.swanliberec.eu) , ktoré ponúkajú rôzne, no prevažne každý rovnaké funkcie od základných technických a servisných údajov bytu (passportu bytu), evidencie nájomníkov až po podvojné účtovníctvo, tlač zloženiek a rozúčtovanie služieb. Aj keď boli všetky programy podobné a poskytovali množstvo funkcií, neponúkali komunikáciu a interakciu s nájomníkom. Maximum, ktoré ponúkali z hľadiska komunikácie s nájomníkmi, bola jedine možnosť odoslania emailu nájomníkovi. Z hľadiska implementácie sú všetky programy koncipované ako spustiteľné aplikácie, ktoré po nainštalovaní a spustení programu poskytnú užívateľovi (prenajímateľovi) dané rozhranie pre pracú s nehnuteľnosťami. V niektorých prípadoch požadujú ešte predinštalovanie databázového systému.

1. Ciele práce

Po analýze výskytu programov tohto typu, som sa rozhodol naprogramovať aplikáciu pre správu nehnuteľností, ktorá by bola určená jak pre prenajímateľov, tak pre nájomníkov. Aplikáciu, ktorá by poskytovala nasledujúce funkcie:

- Okrem základných funkcií pre prenajímateľa aj možnosť komunikácie prenajímateľa s nájomníkmi a nájomníkov s prenajímateľom.
- Nájomníkom by poskytovala informácie o ich ubytovaní, možnosť upravovať osobné údaje a informovať a kontaktovať prenajímateľa o stave nehnuteľnosti priamo z aplikácie.
- Umožniť nájomníkom informovať prenajímateľa o zaplatení jednotlivých predpísaných platieb
- V neposlednom rade aplikáciu, ktorá by nebola viazaná na konkrétny počítač, ale poskytovala by prístup k informáciám priamo z akéhokoľvek počítača, ktorý ma prístup k Internetu.

  Aplikácia by mala poskytovať dve funkčné rozhrania pre jednotlivých užívateľov:

  - Rozhranie pre prenajímateľa
  - Rozhranie pre nájomníkov

  V nasledujúcom texte rozoberiem jednotlivé rozhrania podrobnejšie.

1.1 Rozhranie pre prenajímateľa

Základné rozhranie, ktoré obsahuje nasledujúce sekcie:

- Správa nehnuteľnosti
- Správa nájomníkov
- Financie
- Štatistika

- Úkolovník a pripomienky
- Kontakty
- Správy

Správa nehnuteľností

Úlohou tejto sekcie je poskytnúť funkcie a rozhranie pre definovanie a evidovanie jednotlivých nehnuteľností. Prenajímateľ má mať možnosť zadefinovať si ľubovoľný počet nehnuteľnosti. Hlavná časť sekcie nehnuteľností musí obsahovať základné identifikačné údaje pre danú nehnuteľnosť, ktoré je nutné vyplniť pri jej evidovaní. Prenajímateľ tu musí mať prístup ku všetkým svojím zadefinovaným nehnuteľnostiam.

Správa nájomníkov

Táto sekcia ma na za úlohu poskytnúť funkcie a rozhranie pre prácu s jednotlivými nájomníkmi. Každý evidovaný nájomník musí byť priradený k jednej z evidovaných nehnuteľností. Prenajímateľ tu má prístup k ich osobným údajom a a financiám a aplikácia upozorní prenajímateľa na prípadne dôležité udalosti (nezaplatené nájomné, chýbajúce údaje, ...)

Financie

Funkcie na správu platieb pre jednotlivé nehnuteľnosti a nájomníkov sa nachádzajú v tejto sekcii. Sekcia financie je jedna z najdôležitejších z celej aplikácie. Prenajímateľ tu má možnosť zobraziť a spravovať všetky platby, ktoré má definované a evidované v rámci celej aplikácie, či už sa jedná o platby nájomníkov alebo platby nehnuteľností. Zároveň si tu definuje automatické platby, ktoré budú pravidelne generovať platby pre nájomníkov a nehnuteľnosti.

Štatistiky

V tejto sekcii ma prenajímateľ k dispozícii prehľad príjmov a výdajov na aktuálny, prípadne iný zvolený mesiac. Jednotlivé príjmy a výdaje zahrňujú všetky definované platby v rámci celej aplikácie. Prenajímateľ má možnosť zobraziť si jednotlivé platby, ktoré príjmy tvoria. Ďalej sa tu nachádza interaktívny graf príjmov, v ktorom sa zobrazia názvy a výšky platieb v závislosti na dni splatnosti každej platby.

Úkolovník a pripomienky

Sekcia úkolovník a pripomienky má na starosti správu úloh a pripomienok, ktoré si prenajímateľ zaeviduje v rámci jednotlivých aplikácií, alebo mimo ne. Nachádza sa tu zoznam všetkých úloh, ktoré boli vytvorené. Úkolovník môže slúžiť čisto samotnému prenajímateľovi, alebo sa pri vytváraní úloh môže prenajímateľ rozhodnúť, či sa má daná úloha zobraziť aj nájomníkom, ktorí sú v podnájme v danej nehnuteľnosti. Týmto spôsobom môžu byť nájomníci informovaní o prípadných udalostiach, ktoré nastanú v súvislosti s nehnuteľnosťou (napr. oznámenie o chystaných opravách, o kontrole nehnuteľnosti, odpočtu vodomerov a iné).

Kontakty

V tejto časti aplikácie si môže prenajímateľ evidovať kontakty na svojich dodávateľov, akými sú napr. opravár, upratovacia služba, právnik atď. Jednotlivé kontakty potom môže priradzovať k nehnuteľnostiam. Medzi kontakty môže prenajímateľ zaradiť aj rôzne iné inštitúcie napr. športové centrá a iné strediská, aby svojim nájomníkom uľahčil orientáciu v okolí ich podnájmu.

Správy

Sekcia správy slúži na kontaktovanie a informovanie nájomníkov prenajímateľom. Prostredníctvom správ v rámci aplikácie môže rýchlo oznámiť nové udalosti jednotlivým nájomníkom. Nájomníci majú takisto možnosť kontaktovať prenajímateľa prostredníctvom správ, ktoré sú mu ihneď doručené. Správy v rámci aplikácie slúžia ako osvedčený komunikačný prostriedok a nájomníkom umožňuje viesť si prehľad informácií o nehnuteľnosti, keďže vo svojej schránke správ majú iba správy týkajúce sa nehnuteľnosti, ktorú majú prenajatú.

1.2 Rozhranie pre nájomníkov

Toto rozhranie má slúžiť nájomníkom hlavne ako prístup k informáciám o ich podnájme. Zároveň musia mať možnosť informovať prenajímateľa o stave platieb, ktoré majú predpísané. Rozhranie pre nájomníkov obsahuje nasledujúce sekcie:

- Moje info

- Moje ubytovanie
- Financie
- Správy

Moje info

Táto sekcia tvorí úvodnú časť rozhrania pre nájomníka. Okrem jeho základných osobných údajov sa tu nachádzajú upozornenia na prichádzajúce správy od prenajímateľa a tabuľka s finančným stavom daného nájomníka, ktorá ho upozorní na prípadný dlh.

Financie

Sekcia financií pre nájomníka obsahuje podobné funkcie a informácie ako podkategória financie v rozhraní pre prenajímateľa, no jej úloha je zväčša informatívna. Nájomník tu má zobrazenú tabuľku s automatickými platbami, prostredníctvom ktorých mu aplikácia každý mesiac vygeneruje platby. Nájomník ma možnosť tieto platby zaplatiť a aplikácia následne označí platby ako zaplatené a zobrazí prenajímateľovi informáciu o zaplatení v jeho sekcii financie.

Moje ubytovanie

V tejto sekcii bude mať nájomník podrobný prehľad o svojom ubytovaní. Nájde tu dátum nasťahovania a ukončenia nájmu, počet odbývaných dní a základné údaje o nehnuteľnosti, v ktorej mu je poskytnutý podnájom. Údaje sa získajú od prenajímateľa z jeho rozhrania a podkategórie vybavenie nehnuteľnosti. Voliteľne si môže prenajímateľ nastaviť, či chce, aby sa nájomníkom vypisovali informácie o vybavení nehnuteľnosti. Nájomníkovi sa sem ďalej vypisujú informácie o blížiacich sa udalostiach v súvislosti s platbou, ktoré do aplikácie zadal prenajímateľ z jeho rozhrania. Na tieto udalosti môže priamo reagovať prostredníctvom správ. Z tejto sekcie má nakoniec nájomník prostredníctvom formulára možnosť nahlásiť prenajímateľovi ubytovanie ďalšej osoby. Tento údaj je potrebný pri generovaní evidenčného listu.

Správy

Sekcia správy slúži na kontaktovanie a informovanie prenajímateľa. Prostredníctvom správ v rámci aplikácie majú nájomníci možnosť priamo kontaktovať prenajímateľa o informovať ho, alebo sa ho opýtať na konkrétne detaily.

Evidenčný list

Evidenčný list vygeneruje aplikácia zo zadaných dát od prenajímateľa.

2. Analýza

2.1 Problémy existujúcich implementácií

V súčasnosti sú na českom trhu programy, ktoré sa danej problematike správy nehnuteľností a nájomníkov pomerne dosť komplexne venujú. Ich počet však nie je veľký

a ak by sme sa zamerali na kvalitné programy, našli by sme takých len zopár. Pri prehľadávaní Internetu som narazil na 3 programy, ktoré ma svojimi charakteristikami a obsahom zaujali. Všetko sú to programy, ktoré vyvinuli spoločnosti, ktoré fungujú v oblasti správy nehnuteľností dlhší čas a ich používanie je spoplatnené. (www.jirra.cz, www.des.cz, www.swanliberec.eu)

2.2 Orientácia na zákazníka

Všetky mnou skúmané programy na správu nehnuteľností, pochádzali od firiem, ktoré majú s danou problematikou skúsenosti. To je poznať hlavne z toho, koľko funkcií a informácií jednotlivé programy poskytujú. Hneď na prvý pohľad bolo jasné, že sa orientovali na firmy a inštitúcie, ktoré spravujú veľké počty nehnuteľností (rádovo stovky). Tým pádom sú ich programy preplnené množstvom vstupných formulárov a vyžadujú od užívateľa množstvo údajov. Nevýhodou takýchto programov je, že aj keď sú síce prehľadné, nie sú veľmi zrozumiteľné pre ľudí, ktorí sa tejto oblasti nevenujú. Pre obyčajného človeka, ktorý plánuje prenajímať svoje nehnuteľnosť sa môžu zdať až príliš zložité. Preto som sa rozhodol vytvoriť aplikáciu, ktorá by oslovila práve túto skupinu užívateľov. Ľudí, ktorí majú zopár nehnuteľností a poskytujú ich k prenájmu na určité obdobie.

2.3 Obsah

Dá sa povedať, že obsahovo sú všetky programy na rovnakej úrovni. Každý obsahuje základné funkcie na správu nehnuteľností, ktoré tvorí passport nehnuteľnosti (správa), správa nájomníkov, vyúčtovanie služieb. Ak by si mal človek vybrať po obsahovej stránke, myslím, že by bolo jedno, ktorý z troch programov by si vybral. Jediná časť, ktorá chýba všetkým chýba je prostredie (rozhranie) pre nájomníkov. Či už sa jedná o priame kontaktovanie a komunikáciu prostredníctvom programu, alebo možnosť nájomníka mať prístup k svojim údajom a upravovať ich, nie iba informovať sa.

2.4 Štruktúra a návrh

Táto oblasť ma na programoch zaujímala najviac. Keďže sa jedná o pomerne komplexnú problematiku, jednotlivé firmy mali na výber z dvoch možností, ako svoj program navrhnúť. Buďto vytvoriť jeden kompletný software, ktorý bude navrhnutý ako jedna komplexná aplikácia a na jej správny chod bude potreba mať jej plnú verziu. To znamená, že

ak by sme chceli do aplikácie pridať nejakú ďalšiu časť, potrebovali by sme upgrade celej aplikácie (www.swanliberec.eu, www.jirra.cz). Druhou možnosťou je celý program rozdeliť do modulov (www.destom.cz). Každý modul by sa funkčne venoval inej oblasti (nehnuteľnosti, nájomníci, atď.). Moduly by bolo možné jednoducho pridávať a odoberať podľa potreby (zapínať, vypínať) a bez nutnosti reinštalácie aplikácie.

Nechal som sa inšpirovať práve druhou alternatívou. Je to ideálne riešenie, kedy si sám zákazník na základe vlastných potrieb zvolí, ktorá časť programu je pre neho dôležitá (passport nehnuteľností, nájomné, vyúčtovanie služieb) a ktoré nepotrebuje (prevod bytov do vlastníctva, správa programu). V mojom prípade však nešlo ani tak o zákazníka, ako o programátora, ktorý by chcel aplikáciu rozšíriť i ďalšie funkcie na základe odozvy od zákazníkov. Veľkou výhodou v rozdelenia programu na moduly je, že v rozširovaní programu môže programátor, ktorý tým pádom nebude musieť študovať celý program, stačí mu sa oboznámiť s fungovaním modulov.

2.5 Implementácia

Keďže všetky programy, ktoré sa venujú správe nehnuteľností vznikli pred pár rokmi, sú koncipované ako inštalačný krabicový software. Teda zákazník dostane program, ktorý pred používaním musí nainštalovať. Po inštalácii je program viazaný na konkrétny počítač a môže byť spustený len z neho. Niektoré programy dokonca vyžadujú nainštalovanie databázového servera na daný počítač. Pre zákazníka je takéto riešenie dosť nepohodlné a znamená nevýhodu spravovať nehnuteľnosti z jedného miesta. Táto viazanosť v dnešnej dobre Internetového „boomu“ predstavujú dosť veľké mínus práve pre užívateľov, ktorí očakávajú jednoduchosť, zrozumiteľnosť a nechcú nad programom a správou tráviť veľa času. Aj to je dôsledkom, prečo jednotlivým aplikáciám chýbajú rozhrania pre nájomníkov. U jednotlivých programov by bolo takisto vhodné poskytnúť aspoň nejakú funkčnú trial verziu softwaru, ktorý ponúkajú. Aj keď sa mi podarilo takéto programy nájsť, nepodarilo sa mi ich úspešne nainštalovať. Jedným z hlavných požiadaviek užívateľov je teda jednoduchosť a rýchle spustenie programu (žiadne strácanie času s inštaláciami).

2.6 Hodnotenie a súhrn

Pokiaľ by sa rozhodla nejaká správcovská firma, investovať do kúpy software, ktorý by im pomáhal so správou, určite by si našla vhodný software, ktorý by jej vyhovoval.

Spomínané programy poskytujú naozaj komplexné funkcie a ak s daná firma v oblasti pohybuje, časom by si na daný program zvykla. Jednou nevýhodou pre tieto firmy by bola nutnosť zakúpiť si multilicencie v prípade, že by chcela software inštalovať na viac počítačov a nemožnosť si daný program vyskúšať. Druhou nevýhodou absencia možnosti komunikácie s nájomníkmi a ich nutnosť telefonickej, prípadne osobnej komunikácie.

Na druhej strane, ak by si takýto software chcela zadovážiť nejaká fyzická osoba, ktorá by mala záujem prenajímať svojich pár nehnuteľností, kúpila by si síce software kvalitný, no väčšinu z jeho funkcií by nevyužila. A opäť by chýbala možnosť komunikácie s nájomníkmi, ktorá by v tomto prípade bola ešte dôležitejšia. V tomto prípade by bola vhodnejšia aplikácia, ktorá by umožnila užívateľovi otestovať jej funkčnosť i bez nutnosti akejkoľvek inštalácie.

Ideálnym riešením by bola Internetová aplikácia, ktorá by užívateľom poskytla základné funkcie, komunikáciu nájomníkov a prenajímateľa, pohodlné pridávanie ďalších funkcii a prístup k ktoréhokoľvek počítača. Postupné rozšírenie a pridanie ďalších funkcií do aplikácie by si zvolili užívatelia sami, na základe ich vlastných potrieb.

3. Štruktúra aplikácie

3.1 Návrh riešenia

Z analýzy hotových riešení, ktoré som našiel na Internete, som sa rozhodol, že aplikáciu rozdelím na viacero vzájomne prepojených častí. Každá časť sa bude venovať inej

oblasti (správa nehnuteľností, správa nájomníkov, financie atď.) ale všetky budú medzi sebou vzájomne prepojené. Výhody tejto štruktúry som už spomínal – vzájomne prepojené moduly umožnia ďalej aplikáciu rozširovať, uľahčí sa orientácia v kóde a v rozširovaní aplikácie môže pokračovať aj iný programátor, ktorý nemusí študovať celú štruktúru aplikácie, ale len tú, ktorú potrebuje. Obzvlášť pre Internetové aplikácie je takéto rozdelenie rozumné, kedy nie je potrebné odstaviť celú aplikáciu ak ju upgradujeme.

3.2 Platforma

Keďže som sa rozhodol naprogramovať internetovú aplikáciu, ďalším krokom bola voľba programovacieho jazyka, v ktorom by som aplikáciu na programoval. Na bolo výber niekoľko možnosti. Orientoval som sa však na programovacie jazyky, ktoré patria medzi najznámejšie a u ktorých je predpoklad, že ich ovláda väčšina programátorov. Inými slovami, chcel som, aby aplikácia šla ďalej rozširovať a preto som chcel zvoliť dobre známy jazyk, aby sa programátor, ktorý by ju doplnil, nemusel učiť nový jazyk. Preto som si vyberal z týchto štyroch možností: Microsoft .NET(ASP), PHP, Java(JSF, JSP), Ruby On Rails. (Ešte sa samozrejme ponúka využitie JavaScriptu. Keď sa povie „Internetová aplikácia“, tak pre mňa to implicitne znamená využitie JavaScriptu, preto ho tu nespomínam ako jednu z možností, ale automaticky s ním počítam).

Microsoft .NET (ASP .NET)

.NET ma nejasnú budúcnosť. Microsoft je komerčnou firmou a pokiaľ niečo na trhu zarába málo alebo naopak je na trhu dominantné, tak je jeho ďalší vývoj spomalený, alebo dokonca zastavený. Príkladom je MS Internet Explorer. Hneď keď Microsoft dostal na trhu dominantné postavenie, tak ďalší vývoj bol na niekoľko rokov zastavený. Microsoft ignoroval chybné renderovacie jádro a bezpečnostné nedostatky. Až masívny nástup Gecko/Firefoxu na pole prehliadačov donútil Microsoft pripravovať novú verziu IE 7. To isté sa môže kedykoľvek stať s technológiou .NET

.NET je technológia pre Windows a využíva IIS (web server Microsoftu). Naviac potrebuje pre IIS novú verziu operačného systému (IIS 6 sa nedá nainštalovať na Windows 2000 ani NT). Zvolením tejto technológie by sa automaticky stratila možnosť vyvíjať aplikáciu pod Linuxom, ktorý má vo svete programátorov pomerne slušné zastúpenie.

PHP

PHP sa na Internetovej scéne nachádza pomerne dosť dlho. Našlo si plno priaznivcov a dodnes je celkom atraktívne a hojne sa používa. To, čo ma odradilo od PHP je práca s ním. Pre vývoj aplikácii v jazyku PHP stačí textový editor php procesor, ktorý sa dá zdarma stiahnuť z php.net. To síce nie je zlé, no nevýhoda PHP je, že sa jeho kód kombinuje s kódom HTML. Znepriehľadňuje sa tým zdrojový kód a nepôsobí to na programátora veľmi príjemne. Práve pre túto vlastnosť PHP som si ho nezvolil. Nájdu sa síce nejaké objektové rozšírenia a aj nejaké hotové moduly, ktoré sa dajú v aplikácii použiť, no mňa veľmi neoslovili.

Java (JSF, JSP)

Programovací jazyk Java nepatrí medzi najnovšie a preto za sebou tiahne svoju históriu. Celé prostredie síce bolo globálne vo svojej dobe navrhnuté veľmi dobre, ale vlastná implementácia a štruktúra knižníc už tak geniálna nebola. Knižnice sú naviac neprehľadné, možno dokonca niekde zastaralé a napríklad prístup k súborom je pre nováčikov značne chaotický. Niektoré sú označené ako zastaralé.

Ruby On Rails (RoR)

Framework Ruby On Rails v Českej republike nie je veľmi známy, no vo svete je pomerne dosť rozšírený a veľmi úspešný. Bol vyvinutý, aby uľahčil programátorom budovať a spravovať webové aplikácie. Skladá sa z dvoch častí:

- **Skriptovací jazyk Ruby** - je to jazyk, ktorého autor pri jeho tvorení vychádzal z jazyka Python a Perl. Ukázalo sa, že to bo krok správnym smerom, pretože dodnes má tento jazyk veľký ohlas a úspech. Výhodou tohto jazyka je, že je plne objektovo orientovaný.

- **Framework Rails** – tento framework je kompletne napísaný pomocou jazyka Ruby. Vytvára príjemné štruktúrované prostredie na tvorbu webových aplikácií. Rails je akási kostra celej aplikácie. Ak sa vytvára aplikácia v Rails, každý kód ma svoje miesto a každá časť spolupracuje s každou. Je založený na architektúre MVC – Model, View, Controller, čím dokonale oddeľuje prezentačnú časť aplikácie od jej dátovej. Poskytuje programátorovi vyššiu úroveň abstrakcie pri písaní webových aplikácií, čím uľahčuje prácu hlavne s databázou.

Ruby on Rails poskytuje presne to, čo som od webovej aplikácie požadoval. Dokonale oddeľuje dáta od HTML kódu a umožňuje písať prehľadný kód, ktorý sa dá za krátky čas ľahko pochopiť aj keď sa k programu vrátime po pár mesiacoch. A vďaka MVC architektúre sa dá aplikácia jednoducho rozširovať.

3.3 Architektúra MVC (Model View Controller) v RoR

Aplikácia je teda rozdelená na jednotlivé časti MVC architektúry:

Model

Model je zodpovedný za udržovanie aplikácie v určitom stave. Niekedy je tento stav dočasný, trvá len niekoľko interakcií s užívateľom. Inokedy je stav trvalý a je uložený mimo aplikácie, najčastejšie v databáze. Môžeme si ho predstaviť, ako objekt, ktorý komunikuje s databázou a získava z nej dáta.

Model ale predstavuje viac, než len dáta. Zavádza pravidlá, ktoré sa na ne aplikujú. Napríklad, ak budeme požadovať zvýšenie nájomného nájomníkom, ktorý majú nájomné pod 10 000, tak model sa postará o zavedenie tohto obmedzenia. Implementovaním týchto pravidiel do modelu máme zaručené to, že nič iné v aplikácii nám nemôže znehodnotiť naše dáta. Model sa správa aj ako tzv. gatekeeper – kontroluje prístup k dátam, aj ako úložisko dát.

View (pohľad)

View, alebo pohľad, je zodpovedný za generovanie užívateľského rozhrania, ktoré je obyčajne závislé na dátach z modelu. Príklad: chceme na stránke zobraziť zoznam nehnuteľností, ktoré ma prenajímateľ evidované. Tento zoznam bude prístupný cez model, ale bude to práve pohľad (view), ktorý pristúpi k zoznamu z modelu a výstup naformátuje pre užívateľa. Aj keď pohľady môžu užívateľovi prezentovať dáta v rôznej forme, nikdy nespracúvavajú prichádzajúce dáta. Práca pohľadov skončí v momente, keď sú dáta prezentované užívateľovi. Pre jeden model väčšinou existuje niekoľko pohľadov. Pohľady si môžeme predstaviť ako vygenerované html stránky, xml stránky, alebo templaty pre prácu s JavaScriptom. V jednotlivých stránkach môžeme využiť hniezdený Ruby kód, na generovanie dynamického obsahu. Ide o podobné vkladanie kódu aké ponúka PHP s tým rozdielom, že

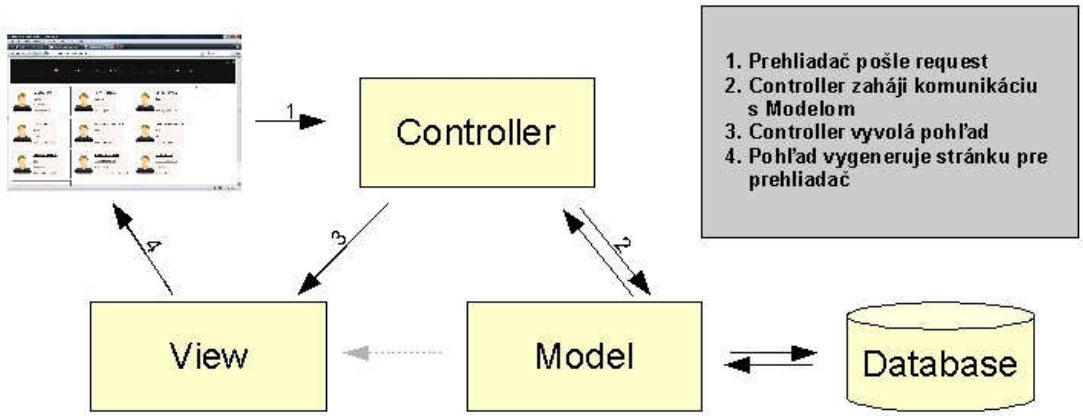
dáta už máme pripravené a iba ich zobrazujeme. V Ruby programátor píše jednotlivé metódy controlleru a modelov.

Controller

Controllery sa správajú ako inštruktori aplikácie. Sú to jej logické centrá. Prijímajú udalosti z vonkajšieho sveta (zvyčajne zo vstupu od užívateľa), spolupracujú s modelom a zobrazia odpovedajúci pohľad. Controllery poskytujú niekoľko pomocných služieb:

- Sú zodpovedné za routovanie externých requestov do interných akcií.
- Poskytujú cacheovanie, ktoré zrýchľuje beh aplikácie
- Poskytujú tzv. Helper moduly, ktoré rozširujú kapacitu šablón pre pohľady
- Spravujú sessiony

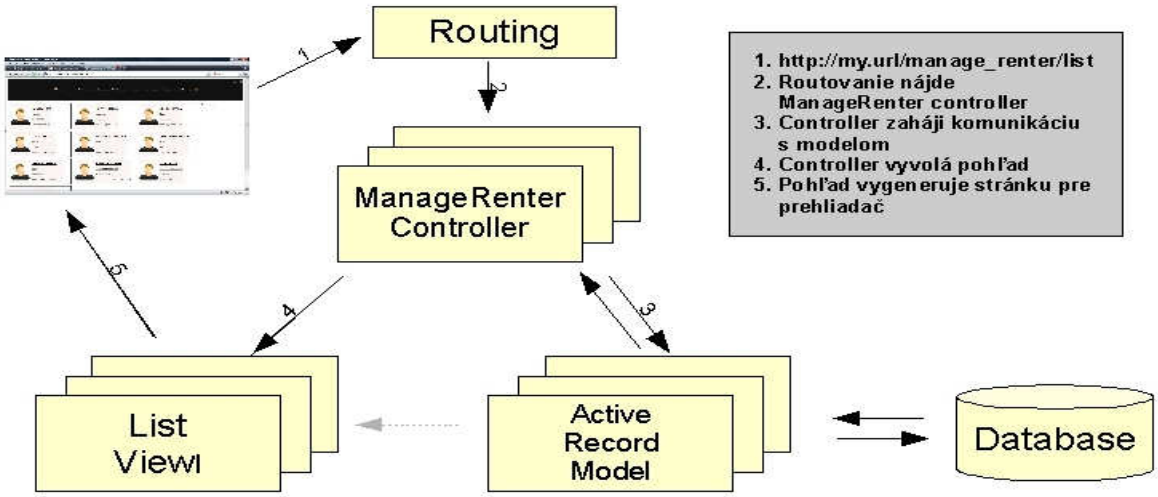
Komunikácia prehliadača s aplikáciou



V Rails aplikácii sú vstupné požiadavky (requesty) najprv poslané do routera, ktorý zistí, kam do aplikácie by mal byť request poslaný a ako ma byť request ako taký rozparsovaný. V tejto fáze sa určí metóda (v Rails nazývaná akciou), ktorá je niekde v controlleri. Akcia sa potom môže pozrieť na dáta v requeste, môže spolupracovať s modelom a môže vyvolať ďalšiu akciu. Nakoniec akcia pripraví informácie pre pohľad, ktorý

vygeneruje nejaký výstup pre užívateľa. Každý controller má svoju vlastnú sadu pohľadov, ktoré obyčajne odpovedajú jednotlivým jeho akciám (metódam).

Spracovanie requestov z prehliadača v aplikácii



Controllery v aplikácii okrem vyššie spomenutých vlastností poskytujú ešte jednu zaujímavú a užitočnú funkciu, ktorou sú filtre. Filtre umožňujú programátorovi napísať kus kódu, ktorý je možné zavolať pred akoukoľvek akciou v samotnom controlleri. Vďaka tejto vlastnosti môžeme napr. pohodlne implementovať autentizačné schéma a prihlasovanie užívateľov. Predtým ako sa zavolá akákoľvek akcia sa najprv zavolá akcia definovaná pre filter a tá zabezpečí všetky potrebné prerekvizity pre ďalší prístup k aplikácii. Do filtru môžeme napr. priradiť akciu na prihlásenie sa užívateľov. Teda k tomu, aby sa užívateľ dostal do aplikácie, filter zabezpečí, že sa musí daný užívateľ prihlásiť. Filtre môžu byť aplikované buď pred zavolaním nejakej akcie, alebo aj po nej.

Helper moduly

Helper moduly sú vygenerované ku každému controlleru. Je to ďalšia šikovná vlastnosť architektúry Rails. Helper moduly sa využívajú vtedy, ak chceme zaviesť určitú abstrakciu nejakých operácii do pohľadov. Do týchto modulov môže programátor umiestniť pomocné formátovacie metódy, ktoré by inak v kóde spôsobovali jeho neprehľadnosť a ktoré funkčne nepatria ani do controllera ani do modelu. Väčšinou sa tieto moduly

používajú na úpravu údajov, ktoré sa zobrazujú pohľadoch resp. ich volanie je možné iba z jednotlivých pohľadov. Pre controller a model nie sú viditeľné (a potrebné).

Sessiony

Session, alebo relácia, umožňuje presnú identifikáciu užívateľa na serveri, ako aj uloženie dát na ňom. Sessiony sa využívajú na komunikáciu medzi serverom a aplikáciou a má k nim prístup iba aplikácia, užívatelia ich nemôžu meniť. Konkrétne Rails uľahčujú prácu s nimi, pretože v programe sa k nim dá pristupovať ako k premenným.

3.4 Rails a trieda Active Record – podpora Modelov

Na reprezentovanie dát z databázy Railsy využívajú ORM - object-relational mapping. ORM knižnice mapujú tabuľky databázy do tried. Ak napr. máme v databáze tabuľku s názvom Renters (nájomníci), v programe budeme mať triedu Renter (nájomník). Jednotlivé riadky z danej tabuľky odpovedajú objektom v programe. Jeden konkrétny nájomník je reprezentovaný ako objekt triedy Renter. V rámci objektu sa používajú atribúty ako get a set metódy, ktoré aktualizujú špecifické bunky v tabuľke. V našom prípade trieda Renter má get a set metódy na nastavenie mena, priezviska, adresy atď. (Samotné Railsy pridávajú do tried metódy na prácu s tabuľkou, akými sú napr. vyhľadanie a uloženie záznamu Renter.find, Renter.save). Rails Teda využíva ORM na mapovanie tabuliek do tried, riadkov do objektov a stĺpcov do atribútov.

Na podporu ORM slúži v Railsoch trieda ActiveRecord. Táto trieda zbavuje programátora starostí s pripojením na databázu, programátor sa prostredníctvom tejto triedy môže plne venovať práci s dátami ako takými. ActiveRecord sa ďalej pohodlne a hladko integruje do zbytku Rails architektúry a poskytuje sofistikovanú validáciu dát, ktoré dokáže extrahovať z webových formulárov a uložiť do modelu. Trieda ActiveRecord dokáže zabezpečiť extrakciu a formátovanie chýb pomocou jedného riadku s kódom. Táto trieda predstavuje dôležitý základ celej Rails MVC architektúry.

Trieda ActiveRecord výrazne uľahčuje prácu s databázou. Neoceniteľnou výhodou, ktorú táto trieda poskytuje vďaka objektovému prístupu, je podpora spojenia medzi databázami. Konkrétne sa v aplikácii nachádza tabuľka Properties (nehnuteľnosti) a tabuľka Renters (nájomníci). Tie majú medzi sebou definovaný vzťah 1 k n, teda jedna nehnuteľnosť

môže mať n nájomníkov a naopak jeden nájomník patrí jednej nehnuteľnosti. Jednej nehnuteľnosti preto, lebo sa vychádza z predpokladu, že v prenajatej nehnuteľnosti je nájomník ubytovaný a teda nemôže byť ubytovaný viac ako v jednej nehnuteľnosti. Ak by sme mali premennú „nehnutelnost“, kde by mal programátor uloženú jednu konkrétnu nehnuteľnosť a potreboval by získať z databáze všetkých jej nájomníkov, jediné, čo mu stačí napísať je: najomnici = nehnutelnost.renters. V premennej nehnuteľnosť potom bude mat pole objektov typu Renter. Je to úžasná vlastnosť triedy ActiveRecord, ktorá výrazne uľahčuje prácu s databázou. Aby tento vzťah fungoval, je treba ho definovať v deklarácii danej triedy Property a Renter.

```
Class Property < ActiveRecord::Base
    # spojenie 1..n:
    has_many :renters
    ...
end
```

```
Class Renter < ActiveRecord::Base
    # spojenie 1..1:
    belongs_to :property
    ...
end
```

4. Rozdelenie a návrh aplikácie

Pri návrhu som dbal na to, aby vytvorená aplikácia bola naprogramovaná z niekoľkých častí, ktoré by medzi sebou komunikovali, ale každý by fungoval nezávisle na druhom. Každá časť aplikácie by potom bola zodpovedná len za svoje dáta, s ktorými by pracovala a nezasahovala by do činnosti ostatných.

Celá aplikácia je preto funkčne rozdelená podľa architektúry MVC. Aplikácia obsahuje spolu 11 sekcií, preto sa skladá z 11 controllerov, ktoré ju ovládajú, 21 modelov, ktoré komunikujú s databázou a pracujú s dátami a niekoľko desiatok pohľadov. Každý controller je zodpovedný za svoju činnosť a nie je závislý od druhého controllera. Ak by sme z aplikácie odstránili nejaký controller, fungovala by ďalej, akurát by mala svoju funkčnosť obmedzenú. Práve týmto spôsobom sa môže aplikácia ďalej rozširovať – pridávaním controllerov (a modelov), ktoré budú mať na starosti nové kategórie správy.

Dôležitým aspektom bola tiež podpora aplikácie, resp. operačný systém, v prostredí ktorého by aplikácia bežala. Keďže sa jedná o webovú aplikáciu, k fungovaniu potrebuje webový server a databázu. Zvolený jazyk RoR podporuje niekoľko webových serverov. Odporúčané sú Apache alebo lighttpd. Ako databázu je možné použiť MySQL, PostgreSQL, SQLite, Oracle, SQL Server, DB2 alebo Firebird. Vďaka tomu je možné spravovať a rozširovať aplikáciu jak pod Windows, tak pod Linuxom či Mac OS.

4.1 Modely aplikácie

Modely v aplikácii predstavujú dáta z jednotlivých tabuliek. Jeden riadok záznamu z tabuľky predstavuje jednu inštanciu triedy v aplikácii. Do jednotlivých tried sú v aplikácii sústreďované všetky operácie z dátami. Tým sa zachová vlastnosti tried ako modelov a z objektového hľadiska sa zachová zapúzdrenosť.

Takmer všetky modely aplikácie reprezentujú údaje z databáze. Ak by sme chceli vložiť nejaké dáta do tabuľky, stačí vytvoriť inštanciu danej triedy, inicializovať ju a uložiť ju zavolaním metódy save. Reprezentácia dát modelami je spôsob prístupu, ktorý používa architektúra RoR.

Zapúzdrenosť bola hlavne dôležitá u triedy Bill, ktorá reprezentuje platbu, ktorú ma zákazník, prípadne prenajímateľ (ak sa jedná o výdaje) zaplatiť. Bill je obsahovo najväčšia

trieda a pracuje s ňou hlavne finančný controller. Pôvodne bola aplikácia navrhnutá tak, že operácie s platením platieb (Bill) mal na starosti práve finančný controller, ktorý dostal od užívateľa pole identifikátorov jednotlivých platieb, prostredníctvom triedy Bill jednotlivé platby získal z databáze, upravil ich na koniec updatoval tabuľku s platbami. Aj keď táto operácia s platbami patrila do tohto controllera, toto riešenie porušovalo vlastnosti zapúzdrenosti triedy Bill, a preto bola všetka logika operácií s platbami a ostatnými dátami presunutá do samotných tried. Tým pádom môžu k operáciám s platbami pristupovať aj ostatné controllery, ak je to nutné.

Jednu z najdôležitejších tried aplikácie tvorí už spomenutá trieda Bill, pretože s ňou aplikácia pracuje najviac a to práve preto, že ňou reprezentuje platby. Spolu s ňou úzko súvisí aj druhá dôležitá trieda aplikácie a tou je trieda Penalty. Tá reprezentuje penále za oneskorené platby. Pri návrhu aplikácie som sa rozhodoval, či penále implementovať ako k platbe vo vzťahu 1..1, teda jedna platba má mať jedno penále alebo umožniť priradiť platbám viac penále. Prípadne vytvoriť jedno penále, ktoré by na základe počtu dni menilo svoju výšku. Rozhodol som sa, že aplikácia má dovoľovať, aby k jednej platbe bolo možné mať priradené viac penále. Myšlienku jedného penále, ktoré by menilo svoji výšku som nerealizoval z dôvodu, že takáto implementácia mi prišlo trochu obmedzujúca a mohlo by dôjsť aj k určitej redundancii penále. Keď by chcel užívateľ priradiť iba jednu výšku penále a to takú, ktorú ma už nejaké existujúce penále vytvorené ale jeho výšku mení na závislosti počtu dní. Užívateľ by tak mal dve penále s tou istou výškou. Miesto toho je penále implementované, ako jednoduché s počtom dni a sumou, ktorá sa má aplikovať. Ak teda chce užívateľ zvýšiť cenu penále, stačí, aby nadefinoval obe penále a pri vytváraní platby ich k nej priradil. Vzťah Bill - Penalty je preto definovaný ako n..n. Z toho dôvodu bolo potrebné vytvoriť ďalšiu tabuľku, ktorá by toto spojenie reprezentovala. Ale s týmto spojením bolo nutné evidovať počet aplikácií daného penále, jeho sumu a príznak, či bola platba zaplatená, alebo nie. Tieto informácie bolo potrebné pripojiť k platbám a zároveň k penále. Ale keďže vzťah Bill Penalty je n..n, tieto informácie nebolo možné uložiť ani do jednej z týchto tabuliek. Zároveň tabuľka, ktorá reprezentuje vzťah n..n má iba dve položky – id platby a id. Vďaka týmto dvom údajom, je možné pristupovať k jednotlivým penále za platby volaním bill.penalties (pridanie príkazu has_many do oboch deklarácii tried Bill, Penalty), ale pridanie ďalších údajov do tejto tabuľky by bolo zbytočné, pretože by sme sa k nim nedostali ani

pomocou triedy Bill, ani Penalty. Tento problém sa vyriešil vytvorením ďalšej triedy AddedPenalty a ta bola deklarovaná, ako trieda, prostredníctvom ktorej je definovaný vzťah n..n medzi Bill a Penalty. Z triedy AddedPenalty sa potom aplikácia dozvie všetky informácie o penále a bude k nej mať z triedy Bill rovnaký pohodlný prístup.

Trieda Bill slúži prenajímateľovi na generovanie platieb pre nájomníkov. Tí však takisto majú mať možnosť evidovať platby ako zaplatené a tým informovať prenajímateľa o prebehnutej platbe. Nemalo by sa ale stať, že by sa v aplikácia objavila evidovaná zaplatená platba, ale v skutočnosti by táto platba zaplatená nebola. Aby nedošlo k tomuto nedorozumeniu medzi nájomníkom a prenajímateľom tým, že by nájomník upravoval priamo platby definované prenajímateľom a zároveň došlo k zachovaniu práv k prenajímateľovým údajom, bolo treba vytvoriť špeciálne platby pre nájomníkov. Tieto platby by mohli nájomníci upravovať ale nemenili by atribúty platieb, ktoré si v aplikácii vytvorili prenajímatelia. Zároveň však nesmie dochádzať k redundancii dát.

Jednou možnosťou, ako umožniť nájomníkom aktívne zasahovať do evidencie platieb bolo umožniť nájomníkom vytvárať platby prostredníctvom triedy Bill, rovnako, ako si ich definujú prenajímatelia. Ale keďže by nesmeli upravovať už existujúce platby od prenajímateľa, musela by sa vytvoriť ich kópia. To by však znamenalo redundanciu dát v tabuľke Bills. Preto bola pre tento účel bola vytvorená trieda TempBill, ktorá reprezentuje zaplatené platby nájomníkom. V databáze je definovaná ako tabuľka, ktorá je referencuje tabuľku s platbami (Bills). Obsahuje ID platby z tabuľky Bills a naviac údaje o dátume platby, výške splátky a penále. Keďže tieto údaje sú redundantné, táto trieda je dočasná, čo naznačuje aj jej názov. Nájomník túto triedu vytvára a prenajímateľ ju ruší. Vytvorením tejto triedy aplikácia oznámi prenajímateľovi informáciu o tom, že nájomník zaplatil predpísanú platbu. V momente, keď túto platbu prenajímateľ potvrdí , aplikácia updatuje príslušnú platbu (Bill) a následne dočasnú platbu zruší (TempBill). Tým pádom môže nájomník aktívne zasahovať do údajov prenajímateľa, ale konečné právo na potvrdenie resp. zaevidovanie prijatej platby prípadne iných údajov má výlučné prenajímateľ. Zachovajú sa tak prístupové práva prenajímateľa a zároveň možnosť nájomníka aktívne zasahovať do evidencie platieb.

Jednou z dôležitých tried je tiež trieda User, ktorá reprezentuje prenajímateľa aj nájomníka ako užívateľov jednotlivých rozhraní. Keďže sa predpokladá, že aplikácia má slúžiť

viacerým užívateľom (prenajímateľom), táto trieda bola vytvorená, aby reprezentovala jednotlivých užívateľov. Obsahuje užívateľské meno a heslo, ktoré je uložene v databáze a zašifrované algoritmom SHA1. Táto trieda je okrem autentizácie využívaná aj ako kontrola pred útokom SQL injection (neoprávnený prístup k dátam a vkladanie dát, ktoré danému užívateľovi nepatria). Pri prihlásení sa užívateľa do aplikácie si aplikácia uloží jeho id do session a pri akomkoľvek získavaní dát z databáze sa najprv overí, či dané dáta patria prihlásenému užívateľovi, prípadne sa základe typu užívateľa vygeneruje príslušný pohľad. Id užívateľa je pridané do tabuľky nehnuteľností. ID prihláseného užívateľa bolo nutné uložiť v aplikácii tak, aby k nej mala prístup iba samotná aplikácia a aby nebolo možné užívateľom toto ID zmeniť. Preto sa pri prihlásení ID prihláseného užívateľa uloží práve do session. Uložením ID užívateľa do session má aplikácia zaručenú dôveryhodnosť tohto údaju, pretože k session ma prístup iba ona sama a užívatelia nie.

V neposlednom rade bolo dôležité umožniť nájomníkom a prenajímateľovi komunikáciu prostredníctvom aplikácie. Ideálnym riešením by sa mohlo zdať implementácia v podobe IM (Instant messaging). Užívateľ by napísal správu a tá by sa hneď zobrazila adresátovi. Na tomto princípe funguje dnes už pomerne dosť rozšírené a obľúbené protokoly typu ICQ, Jabber, MSN, Google talk a iné. Implementácia takejto komunikácie mi však prišla zbytočná hneď z dvoch dôvodov. Tým prvým je samotný existencia už vymenovaných protokolov a ich svetové rozšírenie. Druhým je fakt, že aplikácia na správu nehnuteľností nepatrí medzi internetové aplikácie, u ktorých by užívatelia trávili veľa času. Ide skôr o aplikáciu, ktorá má užívateľov informovať (hlavne nájomníkov). Preto pravdepodobnosť, že by sa prenajímateľ a nájomník stretli online v ten istý čas a zároveň by spolu potrebovali komunikovať je dosť malá. Komunikácia v aplikácii je preto vo forme správ, na ktoré je užívateľ upozornení pri prihlásení sa do aplikácie. Užívatelia majú možnosť si správy prečítať, odpovedať na ne a samozrejme vytvárať a posielať.

4.2 Rozhrania aplikácie

Úlohou aplikácie je možnosť sprístupniť prenajímateľovi a nájomníkom informácie o nájme z akéhokoľvek počítača, bez nutnosti akejkoľvek inštalácie. Preto bola aplikácia navrhnutá ako webová, s dvoma rozhraniami pre prístup – pre prenajímateľa a pre nájomníkov. Funkčne by sa dalo povedať, že rozhranie pre prenajímateľa je administrátorské

a rozhranie pre nájomníka je informačno-komunikačné. Každé rozhranie má preto svoje vlastné controllery, ktoré pokrývajú prácu s jednotlivými sekciami daného rozhrania.

Pre rozhranie pre prenajímateľa sú to controllery, ktoré pokrývajú nasledujúce sekcie:

- Správa nehnuteľnosti (nemovitosti_controller)
- Správa nájomníkov (manage_renter_controller)
- Financie (finances_controller)
- Štatistika (statistics_controller)
- Úkolovník a pripomienky (todos_controller)
- Kontakty (contacts_controller)
- Správy (messages_controller)
- Správa prihlasovacích účtov (login_controller)

Správa nehnuteľností

Súčasťou správy nehnuteľností sú nasledujúce podkategórie:

- **Detail nehnuteľnosti** – hlavná podkategória nehnuteľnosti. Keďže sa jedná o hlavnú podkategóriu, ako prvé sa v tejto sekcii musia zobraziť varovania a informácie vygenerované aplikáciou, ktoré informujú a upozorňujú prenajímateľa o údajoch, ktoré danej nehnuteľnosti chýbajú a musia byť doplnené, aby informácie o nehnuteľnosti boli kompletné. Ďalej sa tu nachádzajú súhrnné informácie o nehnuteľnosti, akými sú názov nehnuteľnosti, adresa, požadované celkové nájomné, poplatky, zoznam nájomníkov(plus odkazy na nich) a stručný popis nehnuteľnosti. Ďalšia funkcia, ktorá sem patrí, je možnosť pridať k nehnuteľnosti poznámky a zadefinovať úlohy, ktoré treba v rámci danej nehnuteľnosti splniť.

- **Vybavenie nehnuteľnosti** – podkategória obsahujúca špecifické informácie a údaje danej nehnuteľnosti. Patria sem údaje o rozlohe nehnuteľnosti ako takej a počet

miestností spolu s ich zoznamom a rozlohami. Ako už názov tejto kategórie naznačuje, prenajímateľ tu bude mať možnosť evidovať si detailný zoznam vybavenia jednotlivých miestností Keďže každá nehnuteľnosť je sama o sebe (niečím) špecifická, musí existovať možnosť doplniť k danej nehnuteľnosti ľubovoľný ďalší údaj. Okrem možnosti pridania ľubovoľných údajov sem patrí špeciálna kategória pridávania informácií o spoplatňovaných službách (elektrina, odpad, daň z nehnuteľnosti a iné) a spoplatňovanom vybavení nehnuteľnosti (vodomery, meracie a regulačné techniky a iné).

- **Financie** – do tejto podkategórie spadajú informácie o všetkých platbách, ktoré súvisia s danou nehnuteľnosťou. Prenajímateľ tu nájde prehľadný zoznam všetkých platieb, ktoré má ďalej možnosť utriediť podľa zvolených kritérií. V zozname je okrem informácie o stave platby (zaplatená, nezaplatená) možnosť danú platbu „zaplatiť“, čo znamená evidovať platbu zaplatenú k danému dátumu. Nejedná sa teda o internetové platby, je to čisto evidencia platieb ako takých. Ku každej nehnuteľnosti existuje možnosť definovať tzv. automatickú platbu. Ide o platbu, ktorá sa pravidelne opakuje a aplikácia generuje jednotlivé platby v závislosti na jej počte dní opakovania. Zoznam všetkých automatických platieb definovaných pre danú nehnuteľnosť nájde prenajímateľ v tejto podkategórii. Jednou z dôležitých funkcií, ktoré sem patria sú informačné údaje o zaplatení konkrétnej platby, ktoré sa k nej vypíšu, ak daný nájomník konkrétnu platbu zaeviduje ako zaplatenú, čo môže učiniť z rozhrania pre nájomníkov.

- **Štatistika** – táto kategória je venovaná finančnej štatistike platieb. Medzi základné informácie tejto podkategórie patrí prehľad príjmov a výdajov pre danú nehnuteľnosť, ktoré predstavujú príjmy a výdaje na aktuálny alebo iný zvolený mesiac. Prenajímateľ tu má ďalej k dispozícii prehľadný interaktívny graf príjmov, v ktorom je zobrazený očakávaný, skutočný a čistý príjem na daný mesiac. Aby informácie o príjmoch boli kompletné, musí byť k dispozícii zoznam platieb, ktoré predstavujú jednotlivé príjmy. Každá z platieb sa graficky znázorní v grafe v závislosti na dni, v ktorom bola zaplatená.

- **Evidenčný list** – z názvu tejto podkategórie vyplýva, čo má byť jej obsahom. Evidenčný list vygeneruje aplikácia zo zadaných dát od užívateľa. Jednotlivé dáta sa

zadávajú pri evidovaní nehnuteľnosti a postupne sa pridávajú ďalšie v sekcii vybavenie v závislostí na konkrétnej nehnuteľnosti.

Všetky údaje nehnuteľnosti má prenajímateľ možnosť upraviť priamo v danej podkategórii.

Správa nájomníkov

Rovnako ako nehnuteľnosti, správa nájomníkov obsahuje nasledujúce podkategórie:

- **Nájomník** – hlavná podkategória správy nájomníkov. Obsahuje základné osobné údaje o danom nájomníkovi, akými sú napr. kontaktná adresa, výška nájmu, dátum začiatku/konca prenájmu, atd. Keďže sa jedná o hlavnú podkategóriu, podobne ako pri nehnuteľnostiach, aj tu sa ako prvé musia zobraziť varovania a informácie, ktoré upozornia prenajímateľa na dôležité udalosti. Okrem základných osobných údajov sa v tejto podkategórii nachádza tabuľka s finančným stavom daného nájomníka, ktorá prenajímateľa upozorní na prípadný dlh nájomníka, a možnosť zaslať danému nájomníkovi správu prostredníctvom aplikácie. Na správy poslané nájomníkom upozorní jednotlivých nájomníkov aplikácia, pri ich prihlásení sa cez rozhranie pre nájomníkov. Dôležitou funkciou je odhlásenie nájomníka, ktorá ukončí nájom konkrétnemu nájomníkovi spolu s vyúčtovaním.

- **Financie** - podkategória obsahujúca rovnaké informácie a funkcie, aké sú k dispozícii v podkategórii financie v sekcii správa nehnuteľností. Platby v tejto kategórii predstavujú platby, ktoré sú viazané na konkrétneho nájomníka (nie priamo na nehnuteľnosť)

- **Štatistika** – opäť sa jedná o finančnú štatistiku, ktorá obsahuje rovnaké funkcie a informácie o príjmoch a výdajoch pre daného nájomníka, aké sú k dispozícii v sekcii správa nehnuteľností

Financie

Jednotlivé podkategórie finančnej sekciu sú:

- **Úvodná stránka –** informuje prenajímateľa o sume a počte platieb, ktoré sú nezaplatené a čakajú na spracovanie a o počte platieb, ktorým vypršala doba splatnosti.

Prenajímateľ ma k dispozícii prehľadný interaktívny koláčový graf, ktorý zobrazuje rozdelenie platieb podľa typu a zobrazí tabuľku jednotlivých platieb.

- **Správa platieb –** podkategória, ktorá obsahuje rovnaké informácie a funkcie ako podkategórie financie v správe nehnuteľností a v správe nájomníkov. Prenajímateľ tu má k dispozícii podrobný prehľad všetkých platieb, ktoré má v evidované v rámci celej aplikácie, nezávisle na nehnuteľnosti a nájomníkovi. Prechodom k tejto podkategórii aplikácia explicitne vygeneruje predpisy na základe všetkých automatických platieb, ktoré ma prenajímateľ definované. Implicitne sú tieto platby vygenerovávané pri prihlásení prenajímateľa do aplikácie.

- **Automatické platby a penále –** v tejto podkategórii má prenajímateľ možnosť definovať si automatické platby, ktoré sa budú pravidelne generovať. Automatické platby sú buď definované pre nehnuteľnosť, alebo pre nájomníka (a tým pádom aj pre nehnuteľnosť). Aplikácia implicitne vygeneruje automatické platby pri prihlásení sa prenajímateľa do nej. Platby sa generujú vždy na aktuálny mesiac. Druhou časťou tejto podkategórie je možnosť prenajímateľa definovať penále za oneskorené platby. Jednotlivé penále je potom možné priradiť k jednorazovým platbám. Jedno penále môže byť priradené k viacerým platbám a jedna platba môže mať priradené väčší počet penále. Aplikácia vygeneruje penále pokiaľ bude prekročený dátum splatností platby, ku ktorej je penále priradené. Prenajímateľ má možnosť penále započítať do platby, alebo jeho konečnú výšku upraviť pri evidovaní platby ako zaplatenej.

- **Príkaz k platbe –** možnosť prenajímateľa vytvoriť príkaz k platbe. Jedná sa o jednorazovú platbu. Pri vytváraní sa definujú jednotlivé údaje pre platbu a prenajímateľ môže k danej platbe pripojiť penále, ktoré si predtým nadefinoval v podkategórii automatické platby a penále. K vytvorenej platba ma potom prenajímateľ prístup v sekcii financie resp. v podkategóriách financie v sekcii správa nehnuteľností, prípadne správa nájomníka v závislosti na príslušnosti platby k nehnuteľnosti alebo nájomníkovi. Pokiaľ je platba viazaná na nájomníka, má k jej údajom prístup aj daný nájomník cez rozhranie pre nájomníkov.

Štatistika

V grafe sa nachádzajú tri príjmy: očakávaný, skutočný a čistý. Očakávaný príjem je príjem, ktorý aplikácia vypočíta na základe automatických platieb, ktoré ma prenajímateľ definované, skutočný príjem je príjem, ktorý obsahuje aj jednorazové platby a platby, ktoré skutočne prebehli a čistý príjem je skutočný príjem od ktorého sú odpočítané výdaje na nehnuteľnosti.

Úkolovník a pripomienky

Sekcia úkolovník a pripomienky má na starosti správu úloh a pripomienok, ktoré si prenajímateľ zaeviduje v rámci jednotlivých aplikácií, alebo mimo ne. Nachádza sa tu zoznam všetkých úloh, ktoré boli vytvorené. Jednotlivé úlohy sa môžu definovať priamo v sekcii správa nehnuteľností v rámci konkrétnej nehnuteľnosti v kategórii detail, alebo z tejto sekcie v prípade, že si chce prenajímateľ vytvoriť pripomienku, ktorú nemá byť viazaná na konkrétnu nehnuteľnosť. Každej úlohe je možné priradiť prioritu, termín splnenia a spôsob hlásenia. Úkolovník môže slúžiť čisto samotnému prenajímateľovi, alebo sa pri vytváraní úloh môže prenajímateľ rozhodnúť, či sa má daná úloha zobraziť aj nájomníkom, ktorí sú v podnájme v danej nehnuteľnosti. Týmto spôsobom môžu byť nájomníci informovaní o prípadných udalostiach, ktoré nastanú v súvislosti s nehnuteľnosťou (napr. oznámenie o chystaných opravách, o kontrole nehnuteľnosti, odpočtu vodomerov a iné).

Aj keď majú prenajímateľ a nájomník spoločnú sekcie financie, bolo nutné vytvoriť pre nájomníkov vlastné controllery pre tieto sekcie, pretože nájomníci nesmú mať rovnaké práva prístupu k jednotlivým údajom. Ich prístup je iba informatívny, nesmú byť schopný priamo meniť dáta, ktoré má prenajímateľ uložené, môžu iba vytvárať nové. Výnimku tvoria osobné údaje nájomníka a posielanie správ.

Controlleri v rozhraní pre nájomníkov pokrývajú sekcie:

- Moje info (renter_controller)
- Moje ubytovanie (renter_controller)
- Financie (finances_controller)

- Správy (messages_controller)

Všetky controllery sú potomkami triedy ApplicationController. ApplicationController je controller, do ktorého sa pridávajú filtre, ktoré by sa mali spustiť na všetkých controlleroch a metódy, ktoré by mali byť dostupné všetkým controllerom. Taktiež je k nemu vytvorený Helper modul, kde sa ukladajú všetky metódy, ku ktorým majú prístup všetky pohľady. Do tohto controllera sú v aplikácia pridané filtre a metódy na autentizáciu užívateľov.

4.3 Controllery rozhrania pre prenajímateľa

Controller financií

Všetky operácie, ktoré súvisia s financiami a jej správou má na starosti tento controller. Patrí sem správa platieb, automatických platieb a penále k platbám. Jeho hlavnou funkciou je vygenerovať platby na aktuálny mesiac podľa predpisu automatických platieb, spracovávať platenie jednotlivých platieb a výpočet penále za oneskorené platby.

Správa nehnuteľností

Controller v tejto sekcii poskytuje prenajímateľovi prístup k správe svojich evidovaných nehnuteľností. Základ všetkých controllorov sú funkcie na podporu pridávania, úpravy a mazania údajov v databáze.

Keďže sa jedná o nehnuteľnosti, controller úzko spolupracuje s triedou Property, ktorá reprezentuje nehnuteľnosť a obsahuje dáta z databázy. Hlavnou úlohou controlleru pre nehnuteľnosti je poskytnúť prenajímateľovi informácie o jeho nehnuteľnosti, ktoré môže upravovať a následne tieto informácie poskytne nájomníkom, ktorí k nim budú mať prístup z ich rozhrania.

Ako je už spomenuté v úvode, sekcia správy nehnuteľnosti ma niekoľko podkategórii. Tieto podkategórie predstavujú v controlleri jednotlivé akcie (akcia = metóda controlleru), ktoré controller zavolá na základe odozvy užívateľa, ktorý s aplikáciou pracuje. Každá akcia ma svoj pohľad, ktorý poskytne prenajímateľovi informácie, ktoré potrebuje.

Controller nehnuteľností, podobne ako controller správy nájomníkov, spolupracuje aj s inými modelmi než len s nehnuteľnosťami. V podkategórii financie sa nachádza správa

financií, kde má prenajímateľ prehľad a správu všetkých financií, ktoré súvisia s danou nehnuteľnosťou. Údaje o financiách dostáva prostredníctvom triedy Bill, ktorý reprezentuje jednotlivé platby. Platby môže prenajímateľ zaplatiť a v tejto kategórii sa k tomu využíva technológia AJAX (technológia je popísaná v programátorskej dokumentácii). Aby sa nemusela cela stránka s platbami obnovovať, AJAX obnoví iba určité elementy na stránke, čím má prenajímateľ väčší prehľad, čo sa s jeho platbami deje. Akcia, ktoré je zavolaná na zaplatenie platieb zistí, či bola zavolaná ako XMLHTTPrequest a podľa toho obnoví buď celú stránku, alebo iba jej časť, ktorej identifikáciu získa zo spomínaného requestu.

Zo sekcie nehnuteľností je pre zadávanie ďalších prídavných informácii venovaná podkategória vybavenie. Tu je možnosť pridať nehnuteľnosti jej špecifické informácie od vybavenia jednotlivých miestností až po služby spojené s nehnuteľnosťou. Dáta z tejto kategórie sú taktiež reprezentované samostatnými triedami a ich hodnoty sú neskôr použité na generovanie evidenčného listu. Popis jednotlivých tried sa nachádza v programátorskej dokumentácii.

Správa nájomníkov

V tomto controlleri sú podobné akcie na správu nájomníkov, ako v controlleri pre nehnuteľnosti. Dôležité funkcie na správu nájomníkov poskytuje samotný controller rovnako ako akcie na prechod k jednotlivým podkategóriám. Controller komunikuje hlavne s triedou Renter.

Obidve sekcie nehnuteľností a nájomníkov sú zároveň miesta, kde je prenajímateľ upozornení na prípadne chýbajúce údaje a na udalosti, spojené s prenájmom. Typicky sa tento kód umiestňuje do pohľadov no v tomto prípade by umiestnenie kódu na generovanie týchto informácií v pohľadoch spôsobilo zavlečenú zložitosť a tým pádom znepriehľadnenie celého kódu. Riešením sú už spomínané Hepler moduly. Tie v tomto miesto poskytli presne tú funkčnosť, ktorú aplikácia potrebovala. Miesto desiatok riadkov kódu v pohľade bol tento kód (na generovanie varovaní) umiestnený do Helperov a v pohľade sa jednoducho zavolá funkcia, ktorá jednotlivé varovania vygeneruje.

Správa nájomníkov poskytuje aj informácie o jednotlivých platbách, ktoré získava od triedy Bill, podobne ako je tomu u finančného controlleru. Platby sa ale týkajú iba konkrétneho nájomníka, podobne to je aj v sekcii s nehnuteľnosťami, ktoré taktiež poskytujú informácie o platbách ale iba v rámci danej nehnuteľnosti.

Controller štatistiky

Okrem zoznamu financií bolo nutné, aby mal užívateľ tiež nejaký celkový finančný prehľad, aký bol príjem jednotlivých nehnuteľností, ich výdavky a podobne. V prípade takýchto prehľadov sú dobrým nástrojom grafy, ktoré umožňujú užívateľovi vytvoriť si lepšiu predstavu o finančnom (a inom) stave.

Controller štatistiky preto reprezentuje údaje o platbách prostredníctvom interaktívneho grafu. Užívateľ má na zobrazenom grafe informácie o príjmoch a výdajoch v rámci daného mesiaca. Pri prechode kurzorom na graf sa v ňom zobrazia príjmy a výdaje na aktuálny deň na základe pozície kurzoru nad grafom.

Controller pripomienok

Dôležitou funkciou, ktorú mala aplikácia poskytovať bolo tiež schopnosť upozorniť prenajímateľa na blížiacu sa udalosť, ktorú si uložil k nehnuteľnostiam, alebo priamo do aplikácie bez akejkoľvek viazanosti na nehnuteľnosť.

Správu jednotlivých udalostí ma za úlohu controller pripomienok, v aplikácii zobrazovaný ako Úkolovník. Prostredníctvom neho umožňuje aplikácia prenajímateľovi poznamenať si k jednotlivým nehnuteľnostiam rôzne úlohy, ktoré treba splniť. Controller potom na základe voľby uloží pripomienky do databáze (prostredníctvom triedy Todo) s príslušným stupňom dôležitosti a príznakom, kde ma aplikácia na danú udalosť prenajímateľa upozorniť.

Controller kontaktov

Prostredníctvom tohto controllera má prenajímateľ možnosť priradiť nehnuteľnostiam kontaktné informácie na dodávateľov (opravár, upratovanie, právnik atď.) Tento controller má význam hlavne pre nájomníkov, pretože ním prenajímateľ poskytne

informácie, na koho sa obrátiť v prípade nejakej nehody a pod. Controller spolupracuje s triedou Contact, ktorá ma na starostí ukladanie kontaktov do databáze.

Controller správ

Tento controller je spoločný pre nájomníkov aj pre prenajímateľa. Zabezpečuje kontakt medzi nimi prostredníctvom rýchlych správ, ktoré sa ukladajú do databázy a sú ihneď po ich odoslaní prístupné adresátovi. Správy v aplikácii sú reprezentované triedou Message.

Login controller

Poskytuje správu užívateľských účtov. Prenajímateľ ma prostredníctvom neho možnosť vytvoriť prihlasovacie meno a heslo pre svojich nájomníkov. Login controller má ďalej na starosti správu prihlasovania do aplikácie a po prihlásení zobrazí prenajímateľovi informačnú tabuľku, kde poskytne dôležite údaje akými sú už spomínané pripomienky, upozorní na blížiace sa ukončenie pobytu nájomníka a na dlžníkov.

4.4 Controllery rozhrania pre nájomníkov

Controller informácií o nájomníkovi

Základný controller pre nájomníkov, ktorý má na starosti informovať nájomníka o jeho osobných údajoch, ktoré poskytol prenajímateľovi a o jeho ubytovaní. Tento controller zároveň umožňuje nájomníkovi zmeniť svoje osobné údaje. Po zmene je prenajímateľ upozornení správou od aplikácie, že prebehla zmena osobných údajov

Controller financií

V rámci zachovania rozhraní som sa rozhodol tento controller urobiť osobitný pre nájomníkov, aby sa v aplikácii neprekrývali jednotlivé finančné operácie nájomníkov a prenajímateľa. Keďže platby a operácie s nimi sú výhradne informácie, ktoré si spravuje sám prenajímateľ, aplikácia neumožňuje nájomníkom priamo zasahovať do ich údajov. Rozhranie, ktoré im poskytuje, je rovnaké, ako pre prenajímateľov ale poskytuje mu iba údaje, ktoré sú pre neho dôležité (predmet platby, deadline nie však typ platby, komu je určená a pod.).

Controller financií nájomníka spolupracuje s triedou TempBill, ktorá reprezentuje zaplatené platby nájomníka a vytvára dočasné platby, ktoré informujú prenajímateľa o zaplatení danej platby. Po ich schválení prenajímateľom, sú z tabuľky vymazané a platba, ku ktorej sa vzťahujú sa zaeviduje ako zaplatená.

5. Programátorská dokumentácia

Celá aplikácia je naprogramovaná v jazyku Ruby On Rails s využitím JavaScriptu a Flashu na zobrazenie grafov v aplikácii. Funkčne je rozdelená na dve časti. Prvá časť je určená pre prenajímateľa resp. vlastníka nehnuteľností. Prenajímateľ tú má možnosť spravovať svoje nehnuteľnosti a nájomníkov, ktorí majú dané nehnuteľnosti prenajaté. Obsahuje sekcie na správu financií, kontaktov a úloh a pripomienok. Druhá časť je venovaná ubytovaným nájomníkom, ktorí tu nájdu informácie o svojom prenájme, financiách majú možnosť kontaktovať a informovať prenajímateľa o prípadných zmenách ohľadom prenájmu.

Vstupné údaje aplikácia prijíma sú od užívateľa z jednotlivých rozhraní vo forme textových reťazcov. Dáta sú následne uložené do databáze a v prípade potreby o ne aplikácia požiada prostredníctvom modelov. Modely majú dáta uložené v jednotlivých triedach, v ktorých sú už pretypovaná na základe typu, v akom sú uložené v databáze.

5.1 Databáza

Dáta celej aplikácie sa ukladajú do MySQL databáze (možné použiť inú). Každá tabuľka je v databáze v nejakom vzťahu (1..1, 1..n, m..n) s aspoň jednou tabuľkou. Tým pádom sa nemôže stať, že v databáze nájdeme údaje, ktoré s ostatnými nesúvisia. Sú v nich uložené základné a osobné údaje. Vybavenie nehnuteľnosti, ako aj ostatné údaje (financie, správy, súvisiace s nimi a nájomníkmi sa nachádzajú vo vlastných tabuľkách, ktoré sú potom k nim pripojené.

Jednotlivé tabuľky majú anglické názvy uvedené v pluráli. Túto konvenciu zavádza architektúra Rails, kvôli lepšej práci s nimi. K jednotlivým údajom v tabuľkách sa potom dostanem cez Model v aplikácii, ktorý má názov v singulári. Ak teda napr. máme tabuľku nájomníkov, jej názov v databáze bude Renters a pristupovať k údajom budeme pomocou Modelu s názvom Renter. Na ukladanie dát do databáze bola zvolená databáza MySQL z toho dôvodu, že je dobre známa a väčšina programátorov s ňou má skúsenosti. Takže pri prípadnom rozširovaní aplikácie by s touto databázou nemali byť problémy.

Nehnuteľnosti a nájomníci sú uložení v samostatných tabuľkách a navzájom referencované vzťahom nehnuteľnoť - nájomník => 1..n. Tento vzťah bol zvolený pre to, že aplikácia predpokladá, že jeden nájomník bude mať prenajatú iba jednu nehnuteľnosť. To aj

preto, že aplikácia je zameraná viac na majiteľov nehnuteľností, u ktorých sa predpokladá, že prenájom nehnuteľností nie je ich hlavná pracovná činnosť. Ide teda o spôsob, ako im zjednodušiť správu prenájmu a tým pádom nepotrebujú veľké množstvo komplexných funkcií.

V tabuľke, v ktorej sú uložené údaje o platbách (tabuľka Bills) sa nachádzajú riadky property_id a renter_id. Keďže platba sa môže patriť buď nehnuteľnosti, alebo aj nehnuteľnosti a zároveň aj nájomníkovi. Nemôže sa stať, že nejaká platba bude patriť iba nájomníkovi a nie nehnuteľnosti, pretože sa predpokladá, že ak je nejakému nájomníkovi platba priradená, tak nájomník je ubytovaný v nejakej nehnuteľnosti. Môže sa potom zdať, že výskyt oboch riadkov (property_id, renter_id) je zbytočný a je teda efektívnejšie nahradiť ich jedným riadkom id a riadkom typ, ktorý by bol typu ENUM(property, renter). Zvolené riadky property_id a renter_id boli z toho dôvodu, že môže dôjsť k presťahovaniu nájomníka do inej nehnuteľnosti. Môže nastať situácia, že by sme mali záznam o platbe, ktorý by mal id nájomníka XY a daný nájomník by sa presťahoval do inej nehnuteľnosti. Tým pádom by sme stratili údaj o tom, ktorej nehnuteľnosti patrí daná platba. Či patrí do starej nehnuteľnosti, alebo do tej, kde sa nájomník práve býva. Druhým dôvodom, prečo je výhodnejšie mať oba riadky je prípad, kedy by sa aplikácia upravila tak, aby podporovala prenajatie viac ako jednej nehnuteľnosti jednému nájomníkovi. Jasne sa tým odlíši príslušnosť platby k nehnuteľnosti, ak bude mať nájomník prenajatú viac nehnuteľností.

5.2 Modely

Ako už bolo niekoľko krát spomenuté, jednotlivé modely reprezentujú údaje v databáze. Databáza obsahuje 19 tabuliek s údajmi. Jednotlivé modely aplikácie sú:

- **Accesorry –** reprezentuje príslušenstvo nehnuteľnosti, ktoré tvorí jeho stálu súčasť a je účtované jeho používanie. Trieda Accessory viazaná na nehnuteľnosť a v aplikácii sa jej údaje využíva na generovanie evidenčného listu.

- **AccommodationHistory –** trieda slúži na vytváranie histórii ubytovania. Aplikácia vždy pri nasťahovaní nového nájomníka do nejakej nehnuteľnosti vytvorí záznam v databáze o novom ubytovaní, kde zaeviduje nájomníka, nehnuteľnosť a dátum odkedy

dokedy daný trvá nové ubytovanie. Užívateľ tak má prehľad, kto bol kedy a kde ubytovaní.

- **AddedPenalty -** prostredníctvom tejto triedy je v aplikácii definovaný vzťah n..n medzi triedami Bill a Penalty. Trieda generuje počet penále, ktorý má byť pripočítaný k platbe, ktorej vypršala doba splatnosti a históriu pripočítania penále. Z údajov o penále zistí, po koľkých dňoch sa penále aplikuje a v závislosti na aktuálnom dni vypočíta, koľkokrát sa má dané penále aplikovať a údaj o počte uloží do databáze. Penále je potom zobrazené pri danej platby v sekcii financie. Výsledné penále ma prenajímateľ možnosť upraviť a po zaplatení, sa výška penále uloží do tabuľky AddedPenalties, a trieda AddedPenalty vygeneruje históriu pridelenia jednotlivých penále, aby mal prenajímateľ prehľad o pripočítaných penále.

- **Additional –** trieda reprezentuje údaje nehnuteľnosti a nájomníka, ktoré má k nim prenajímateľ možnosť definovať naviac. Dáta sú uložené v jednej tabuľke, pretože majú rovnaké atribúty. Jediný rozdiel je ich príslušnosť k nehnuteľnosti alebo nájomníkovi. Tú reprezentuje výčtový typ ENUM v tabuľke, podľa ktorého aplikácia zistí, komu údaj patrí.

- **AutoPay –** automatické platby, ktoré si zadefinuje prenajímateľ viazané na nehnuteľnosť alebo nájomníka reprezentuje táto trieda. V tabuľke AutoPays sú uložené jednotlivé definície automatických platieb a z ich údajov sa vygenerujú platby. Na základe údaju o počte dní opakovania a dátumu posledného generovania vygeneruje platby do konca aktuálneho mesiaca. Platby sú reprezentované triedou Bill.

- **Bill –** jednotlivé platby v aplikácii reprezentuje táto trieda. Pomocou nej získava prenajímateľ informácie o platbách, umožňuje mu evidovať platby ako zaplatené v sekcii financie. Platbu môže evidovať ako čiastočne zaplatenú, alebo plne zaplatenú. Pri čiastočnej platbe sa vytvorí nová platba, ktorá má poznámku, že je čiastočná a cena pôvodnej platby je znížená. Pôvodnej platbe je taktiež pridelená poznámka, že bola čiastočne zaplatená. Trieda Bill je ďalej zodpovedná za výpočet penále, na ktorom spolupracuje s triedou AddedPenalty. Tá poskytuje výpočet počtu penále a trieda Bill ďalej na základe tohto počtu toto penále. Trieda Bill získava informácie o platbách a na základe nich vypočítava príjmy a výdaje. S týmito údajmi ďalej pracuje controller štatistiky. Controlleru trieda poskytuje dáta na zobrazovanie grafu v sekcii štatistika.

- **Contact -** trieda reprezentujúca kontakty prenajímateľa. Aplikácia ju využíva na vytváranie a ukladanie kontaktov k jednotlivým nehnuteľnostiam.

- **Fitting –** slúži na evidovanie vybavenia (nábytku) jednotlivých nehnuteľností.

- **Message –** trieda reprezentuje správy, prostredníctvom ktorých medzi sebou komunikujú prenajímateľ a nájomníci. Prenajímateľovi má možnosť posielať správy jednotlivým nájomníkom. Nájomníkom umožňuje posielať správy prenajímateľovi.

- **MyDate -** jedna z tried, ktorá nie je viazaná na tabuľku v databáze. Táto trieda slúži na formátovanie časových údajov a dátumov. Časové údaje, ktoré sa ukladajú do databázy, ako napríklad deadline splatnosti nejakej platby sú ukladané v sekundách. Pomocou tejto triedy sa uložené sekundy prevedú na dátum a čas.

- **Note –** umožňuje priradiť poznámky k platbám a nehnuteľnostiam

- **PenaltyHistory –** ukladá do databázy informácie o pripočítanom penále k jednotlivým platbám. Každý záznam obsahuje údaj o pôvodnej výške a novej pripočítanej výške penále.

- **Penalty –** reprezentuje jednotlivé penále, ktoré si užívateľ v aplikácii nadefinuje

- **Property –** slúži na prácu s nehnuteľnosťou. Táto trieda sa v aplikácie využíva na overenie príslušnosti získaných údajov prihlásenému užívateľovi. Keďže takmer všetky dáta z tabuliek v databáze, sú prepojené s touto triedou, k tabuľke Properties (nehnuteľnosti) je pridaný atribút user_id, ktorý reprezentuje jej príslušnosť k užívateľovi. Preto predtým, než sa akékoľvek dáta zobrazia užívateľovi, sú najprv testované, či skutočne patria prihlásenému užívateľovi. Táto ochrana sa využíva v prípade, že by sa prihlásený užívateľ snažil získať dáta, ktoré mu nepatria modifikovaním zdrojových kódov HTML stránky.

- **Rent –** trieda reprezentujúca jednotlivé nájomné nájomníkov. Využíva sa na uloženie informácii o zmenách nájomného v priebehu ubytovania. Každý nájomník má svoje nájomné uložené v tabuľke Rents a pri zvýšení, prípadne znížení sa vytvorí nový záznam. Staré nájomné v tabuľke ostáva spolu s novým, tým pádom má užívateľ k dispozícii históriu zmeny nájomného.

- **Renter -** trieda reprezentujúca nájomníka nehnuteľnosti. Okrem získavania základných údajov nájomníka z databáze sa využíva na výpočet príjmov a výdajov daného nájomníka a výpočet dlžnej čiastky.

- **Room –** reprezentuje dáta z tabuľky Rooms, kde sú uložené názvy základných izieb v nehnuteľnosti akými sú kuchyňa, kúpeľňa, balkón a iné. K jednotlivým nehnuteľnostiam sú potom tieto izby referencované vzťahom n..n. Každá izba má údaje o svojej rozlohe. Údaje o rozlohe má na starosti trieda RoomArea.

- **RoomArea –** táto trieda je použitá na prepojenie tabuliek Properties a Rooms, ktoré majú medzi sebou vzťah n..n. Vyžíva sa na pridanie informácii o rozlohe k danej izbe.

- **Todo -** trieda reprezentuje úlohy a pripomienky, ktoré sú priradené k jednotlivým nehnuteľnostiam. Aplikácia prostredníctvom tejto triedy umožňuje upozorniť užívateľa na blížiacu sa udalosť a nutnosť jej splnenia.

- **User -** pre každého užívateľa, ktorý si zadefinuje nejaké nehnuteľnosti v aplikácii, je v databáze vytvorený záznam s jedinečným id, ktoré je pridané ku každému záznamu k jeho nehnuteľnostiam. Trieda potom prostredníctvom tohto id a faktu, že všetky údaje (nájomníci, vybavenie nehnuteľnosti, platby, penále atď.) sú viazané na nejakú nehnuteľnosť, získava z databáze iba údaje, ktoré patria prihlásenému užívateľovi. Zaručuje, že sa jednotlivým užívateľom budú zobrazovať iba dáta, ktoré si tam sami vložili. Trieda ďalej poskytuje autorizáciu užívateľa pri prihlásení. Pri vytvorení nového užívateľa jeho heslo zahashuje do databáze algoritmom SHA1 a pri prihlásení overí jednotlivé hashe hesla.

5.3 Controllery aplikácie

Controller financií

Všetky operácie, ktoré súvisia s financiami a jej správou má na starosti tento controller. Patrí sem správa platieb, automatických platieb a penále k platbám. Jeho hlavnou funkciou je vygenerovať platby na aktuálny mesiac podľa predpisu automatických platieb, spracovávať platenie jednotlivých platieb a spúšťanie výpočtu penále za oneskorené platby.

Generovanie automatických platieb spustí controller prostredníctvom triedy AutoPay. Tá reprezentuje automatickú platbu a jednotlivé automatické platby generuje vždy na aktuálny mesiac podľa počtu dní opakovania danej automatickej platby. Po tejto generácii už má aplikácia k dispozícii platby, z ktorých môže vypočítať očakávaný príjem, informácie o ktorom ma prenajímateľ v sekcii štatistika.

Výpočet penále prebieha prostredníctvom triedy AddedPenalty, cez ktorú je v aplikácii definovaný vzťah n..n medzi triedami Bill a Penalty. Táto trieda zároveň slúži na generovanie histórii penále, z ktorých sa prenajímateľ dozvie v ktorý deň aká hodnota penále bola pripočítaná. Celkové pridané penále má však prenajímateľ možnosť pred započítaním zmeniť.

Controller ďalej zobrazuje užívateľovi jednotlivé platby. Prostredníctvom triedy User získa všetky platby z databáze, ktorá ich vyhľadá v prostredníctvom id prihláseného užívateľa, ktoré je uložené v session pri prihlásení. Spravovať (vytvoriť, zaplatiť, upraviť, zrušiť) jednotlivé platby nehnuteľností a nájomníkov je možné prostredníctvom triedy Bill.

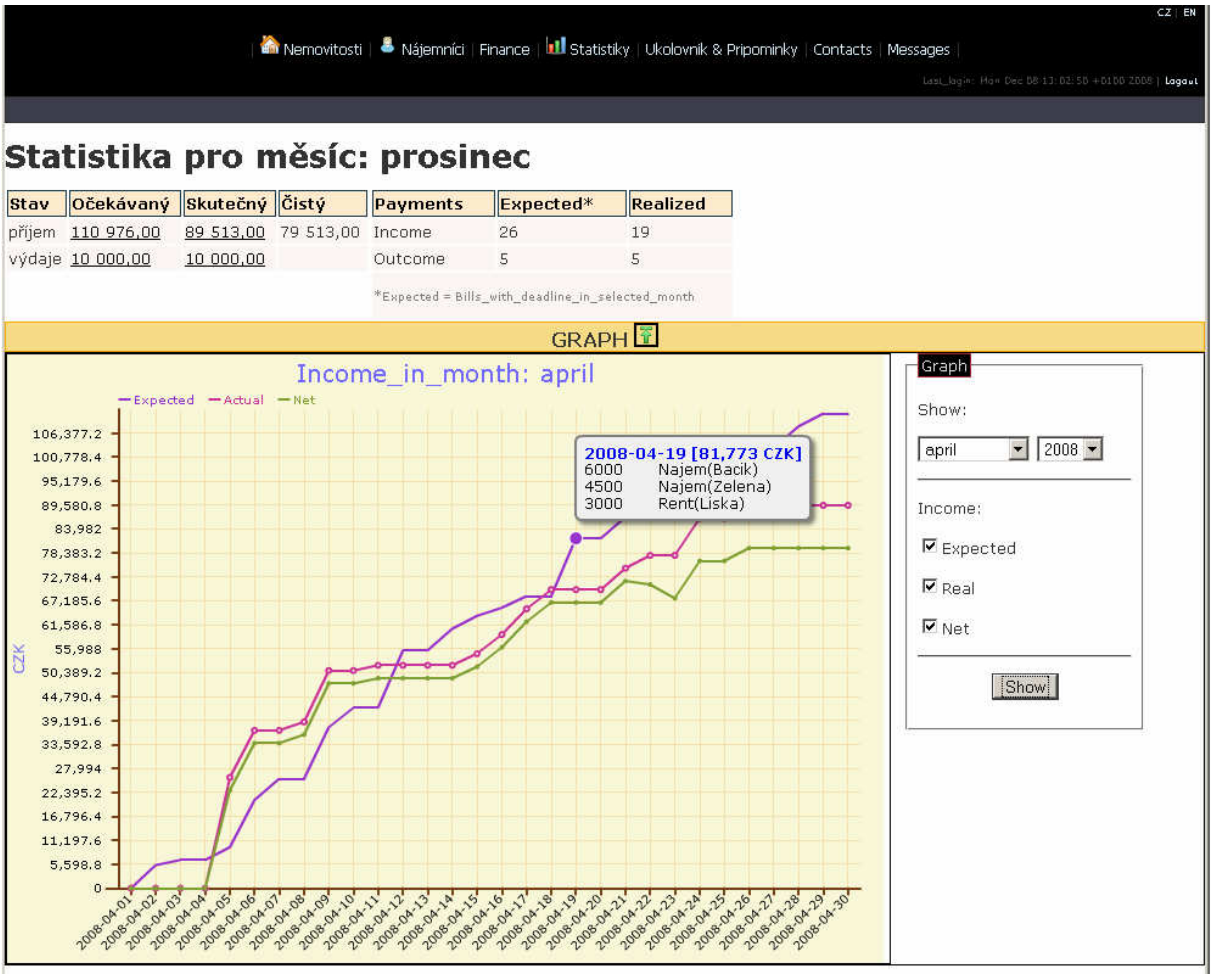
Controller štatistiky

Controller doplňuje informácie o financiách. Užívateľovi zobrazí graf príjmov za odpovedajúcu mesiac. Údaje získa prostredníctvom triedy Bill, ktorá controlleru zároveň údaje predá v správnom formáte. Zo získaných údajov vyberie tie, ktoré sú potrebné pre zobrazenie príjmu (výška platby, stav platby, kategória platby) a následne daný príjem zobrazí vo forme grafu.

Na vykreslenie grafu v aplikácii, ktorý by zobrazil výšku príjmu v závislosti na dni v mesiaci sa použil plugin do RoR (http://www.pullmonkey.com/projects/open_flash_chart), ktorý je voľne dostupný a je poskytuje niekoľko typov prehľadných grafov, ktoré sú naviac interaktívne. Grafu získa potrebné informácie o platbách a umožňuje nastaviť názov jednotlivých ôs, takže užívateľ má pred sebou zrozumiteľný prehľad svojich príjmov.

Graf je implementovaný pomocou jazyka ActiveX a Flashu. Práve vďaka Flashu pôsobí na užívateľa príjemným dojmom. Aby sa pohodlie práce s grafom ešte zvýšilo, aplikácia opäť využíva technológiu AJAX, na jeho zobrazenie a update údajov p príjmoch.

Sekcia štatistiky financií v rozhraní pre prenajímateľa s grafom



Controller nehnuteľností

Controller nehnuteľností pracuje hlavne s triedou Property, ktorá ich reprezentuje. Do tohto controllera je sústredená všetka logika s operáciami okolo nehnuteľnosti. Okrem základných operácii s pridávaním, upravovaním a mazaním nehnuteľností má tento controller na starosti aj správu ďalších údajov, ktoré užívateľ pridá k danej nehnuteľnosti. Jedná sa hlavne o pridávanie informácií o vybavenosti nehnuteľnosti (Trieda Fitting), informácií o poskytovaných službách pre danú nehnuteľnosť (Trieda Accessory) a pridávanie jednoduchých textových informácií o danej nehnuteľnosti (trieda Additional).

Ďalšou triedou s ktorou controller nehnuteľností spolupracuje je trieda RoomArea. Táto trieda umožňuje užívateľovi pridať k nehnuteľnosti izby spolu s ich rozlohami. Tieto údaje sa ďalej využívajú pri generovaní evidenčného listu nehnuteľnosti. Izby sa implicitne vytvoria s nulovými rozlohami pri definovaní novej nehnuteľnosti podľa údaju o počte izieb.

Každá nehnuteľnosť poskytuje užívateľovi históriu ubytovaných nájomníkov. Na generovanie histórie controller využíva triedu AccommodationHistory, ktorá získa z databáze všetky potrebné údaje.

Ak sa užívateľ rozhodne danú nehnuteľnosť zavrieť, controller túto operáciu vykoná v spolupráci s triedou Property. Trieda property najprv vykoná potrebné operácie s nájomníkmi, ktorí sú v nej ubytovaní. Prenajímateľ má možnosť vybrať, aká operácia sa má s nájomníkmi vykonať, pred uzavretím nehnuteľnosti. Buď sa im uzavrie nájom, alebo je možné ich presťahovať do inej nehnuteľnosti, alebo ostanú v danej nehnuteľnosti. Na základe voľby prenajímateľa trieda Property spracuje požiadavky na nájomníkov. Ak sú k danej nehnuteľnosti definované automatické platby, prenajímateľ má opäť možnosť označiť jednotlivé platby a trieda Property následne zastaví ich generáciu.

Controller nájomníkov

Podobne ako controller nehnuteľností, aj controller nájomníkov poskytuje základne funkcie pre správu nájomníkov. Controller spolupracuje hlavne s triedou Renter, ktorá reprezentuje nájomníka. Tá ďalej využíva triedy AccommodationHistory a Rent v prípade, že užívateľ chce presťahovať nájomníka, alebo mu zmeniť výšku nájomného.

Controller pri zmene nájomného zavolá metódu triedy Renter, ktorá zmenu prevedie. Keďže nájom je pravidelná platba, aplikácia umožňuje prenajímateľovi rovno upraviť automatickú platbu, ak je pre dané nájomné definovaná. Túto vlastnosť zabezpečujú triedy Renter, Rent a AutoPay. Trieda Renter zistí o ktoré nájomné (aktuálne) sa jedná, keďže nájomník ich môže mať viac, ak už raz bolo nájomné upravené. Trieda Rent následne overí, či má dané nájomné definovanú automatickú platbu. V prípade existencie automatickej platby pre nájomne sa na základe dátumu, od ktorého nové nájomné platí, daná platba upraví. Zároveň aplikácia ponúkne užívateľovi možnosť vymazať už vygenerované platby, ktoré majú staré nájomné a dátum ich splatnosti je v budúcnosti. Ak sa nájomné mení k dátumu, ktorý je v minulosti, automatickej platbe sa zmení dátum posledného generovania a výška nájomného. V prípade, že sa nájomné bude meniť v budúcnosti, danému nájomnému sa vytvorí nová automatická platba. Zároveň sa však aktuálnej automatickej platbe upraví dátum, kedy skončí jej generovanie, na deň, od ktorého začne nová automatická platba generovať platby za nájomné. Užívateľ má k dispozícii históriu automatických platieb.

V prípade, že sa užívateľ rozhodne presťahovať nájomníka, presťahovanie zabezpečí trieda Renter. Tá spolupracuje s triedou AccommodationHistory, ktorá pracuje s informáciami o jednotlivých ubytovaniach. Pri presťahovaní nájomníka trieda AccommodationHistory vytvorí nový záznam o ubytovaní nájomníka. Prenajímateľ má tak prehľad o kompletnej histórii ubytovaných nájomníkov, ku ktorej má prístup v sekcii správa nehnuteľností.

Controller úloh a pripomienok

Controller má na starosti správu úloh a pripomienok, ku ktorej má užívateľ prístup v sekcii úkolovnik, alebo v rámci danej nehnuteľnosti, ktorá má definovanú nejakú úlohu alebo pripomienku. V úkolovníku ma prenajímateľ prehľad všetkých úloh, ktoré si zadefinoval.

Jednotlivé úlohy a pripomienky má prenajímateľ možnosť definovať ku každej nehnuteľnosti. Controller pri správe úloh pracuje s triedou Todo, ktorá ich reprezentuje. Trieda Todo umožňuje užívateľovi definovať úlohu a určiť, či sa daná úloha má zobraziť aj nájomníkom, alebo je to iba jeho súkromná pripomienka. Controller na základe týchto údajov zobrazí jednotlivé úlohy prenajímateľovi a nájomníkom.

Controller kontaktov

Controller slúži na správu kontaktov pre jednotlivé nehnuteľnosti. Využíva triedu Contact. Priradenie jednotlivých kontaktov k nehnuteľnostiam prostredníctvom tejto triedy má na starosti controller nehnuteľností.

Controller správ

Tento controller spolu s triedou Message zabezpečuje komunikáciu medzi prenajímateľom a nájomníkmi. Jednotlivé správy sa ukladajú do databáze s príznakom, či už boli prečítané, alebo nie. Aplikácia užívateľom oznámi počet nových správ a controller správ im potom umožní k jednotlivým správam pristupovať. Prenajímateľ má možnosť poslať správu ktorémukoľvek nájomníkovi. Nájomníci však môžu posielať správy iba prenajímateľovi, nie svojim spolunájomníkom. Predpokladá sa, že to nie je nutné, keďže všetci nájomníci sú ubytovaný v rovnakej nehnuteľnosti.

Controller prihlasovania

Controller ma na starosti proces prihlasovania užívateľov do aplikácie. Spolupracuje s triedou User, ktorá reprezentuje tých užívateľov aplikácie, ktorý majú vytvorené prihlasovacie meno a heslo. Pri prihlásení ukladá id užívateľa do session. Pri akomkoľvek zobrazovaní údajov z aplikácie potom toto id slúži na overenie údajov, ktoré si užívateľ vyžiadal od aplikácie.

Trieda User ma za úlohu autentizáciu užívateľov. Heslá jednotlivých užívateľov sú pred uložením do databáze zahašované hashovacím algoritmom SHA1. Pri prihlásení užívateľa trieda User porovná jednotlivé hashe hesiel a ak sa zhodujú, užívateľovi je poskytnutý prístup do aplikácie.

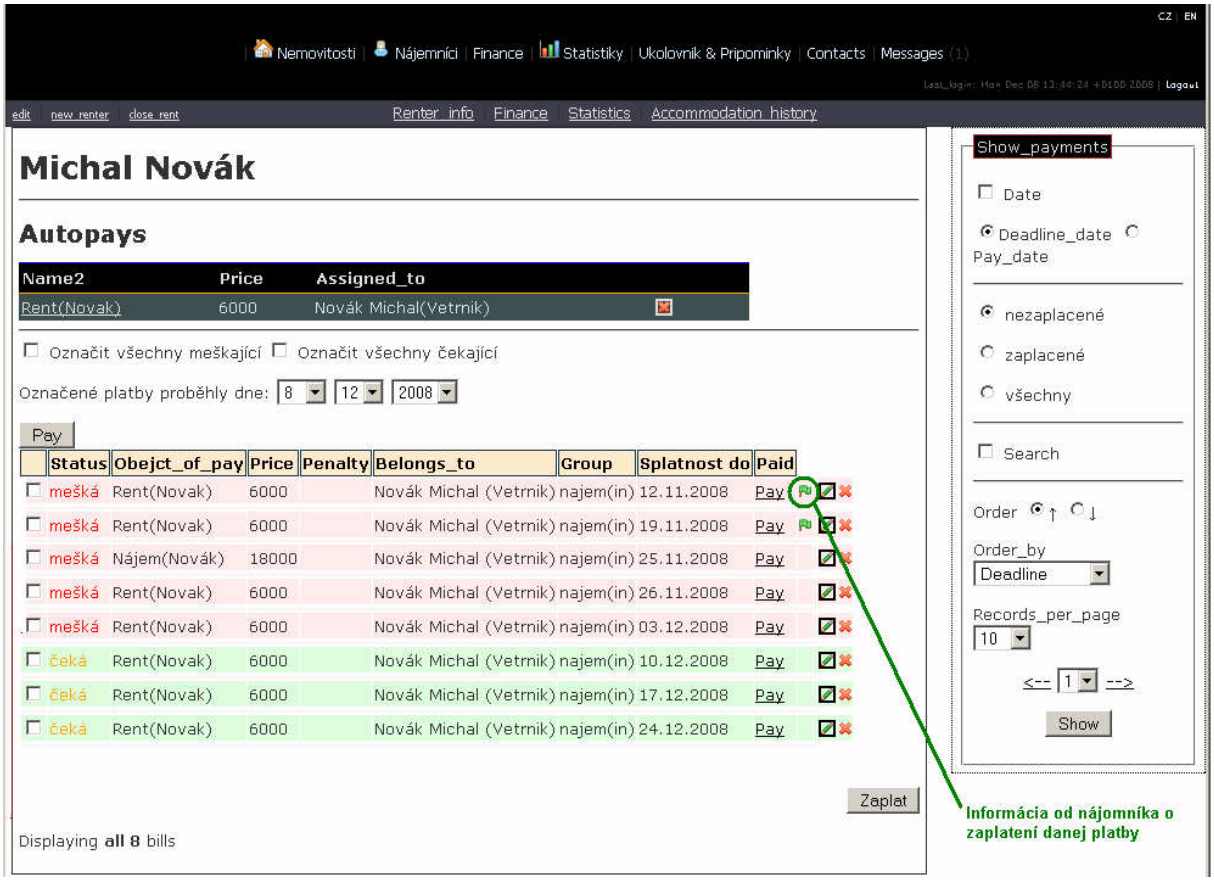
Vytváranie jednotlivých účtov pre nájomníkov má na starosti sám prenajímateľ. Po definovaní jednotlivých nájomníkov k nehnuteľnostiam ho aplikácia upozorní na užívateľov, ktorí nemajú vytvorené prihlasovacie meno a heslo. Prenajímateľ potom má možnosť dané prihlasovacie údaje vytvoriť z rozhrania pre nájomníkov. Prihlasovacie údaje vytvorí trieda User.

Controller financií pre nájomníkov

Tento controller bol vytvorený pre nájomníkov, aby mali možnosť informovať prenajímateľa. Spolupracuje hlavne s triedou TempBill, prostredníctvom ktorej aplikácia zaznamenáva platby nájomníkov.

Controller zobrazí nájomníkom predpísané platby prostredníctvom triedy Bill. Nájomníci majú následne možnosť dané platby zaplatiť. Needitujú pritom však priamo predpísané platby, ale prostredníctvom triedy TempBill vytvoria dočasné platby. Vďaka triede TempBill potom controller financií pre prenajímateľa upozorní daného prenajímateľa na zaplatené platby. Ten má potom možnosť danú platbu potvrdiť, alebo počkať a overiť si, že platba bola skutočne zaplatená.

Sekcia financií rozhrania pre prenajímateľa



Controller informácií pre nájomníka

Controller informácií umožňuje nájomníkom zobraziť si informácie o aktuálnom ubytovaní. Umožňuje im taktiež upravovať ich osobné údaje a tým informovať prenajímateľa, pokiaľ došlo k nejakej zmene.

Ďalšou funkciou, ktorú poskytuje tento controller nájomníkom je nahlásenie novej osoby, ktorá sa nasťahovala do danej nehnuteľnosti. Nájomníci majú možnosť z ich rozhrania vyplniť osobné údaje novej osoby a následne vytvoriť a priradiť danú osobu k nehnuteľnosti, ktorú majú prenajatú. Vytvorenie novej osoby sa vykoná prostredníctvom triedy Renter. Controller po vytvorení a priradení nového nájomníka k nehnuteľnosti vytvorí a pošle správu prenajímateľovi, kde ho o tejto udalosti informuje.

Záver

Cieľom tejto bakalárskej práce bolo vytvoriť aplikáciu pre majiteľov a prenajímateľov nehnuteľností, ktorým by uľahčila prácu pri správe týchto nehnuteľností. Okrem správy nehnuteľností pre nájomníkov mala aplikácia poskytovať aj rozhranie pre nájomníkov, ktorí by tak mali prístup k aktuálnym informáciám o svojom prenájme a mohli kontaktovať svojho prenajímateľa.

Najčastejšou a dôležitou oblasťou pri správe nehnuteľností je správa financií. V aplikácii som sa snažil spracovať túto oblasť tak, aby užívatelia mali pohodlný prístup k informáciám o platbách a aby aplikácia bola schopná sama vygenerovať platby za nájom a iné služby. Tým pádom by jednotlivý užívatelia iba evidovali splátky jednotlivých platieb. Aj keď sa mi tento zámer podaril splniť a vytvoriť prehľadné informácie o platbách a ich štatistikách, daná problematika správy platieb je viac komplexnejšia, než sa na prvý pohľad zdá. Aby bola táto správa kompletná, je nutná hlbšia znalosť finančného účtovníctva. Vytvorenie kompletného účtovníctva však nebolo mojím hlavným cieľom. Účtovníctvo preto patrí medzi jedno z možných budúcich rozšírení aplikácie.

Druhým dôležitým cieľom bolo vytvoriť v aplikácii rozhranie pre nájomníkov. Mojou snahou nebolo vytvoriť aplikáciu, u ktorej by jednotlivý užívatelia – nájomníci trávili veľa času. Bolo dôležité, aby mali rýchly a jednoduchý prístup k svojim informáciám a možnosť určité informácie editovať. Preto dané rozhranie poskytuje nájomníkom základné funkcie na úpravu osobných údajov, platenie platieb a nahlásenie novej osoby v nehnuteľnosti. V neposlednom rade bolo taktiež dôležité umožniť nájomníkom prostredníctvom aplikácie kontaktovať prenajímateľa. Komunikácia bola zvolená prostredníctvom správ, ktoré sa ukladajú do databáze a daný adresát je na ne upozornený. Vzhľadom k typu aplikácie sa zvolený spôsob komunikácie ukázal ako postačujúci. Myslím, že rozhranie pre nájomníkov splňuje požiadavky, ktoré som na ne kládol pri stanovení cieľov aplikácie.

Pre prenajímateľov aplikácia poskytuje funkcie pre pohodlnú správu nehnuteľností a ich nájomníkov. Majú možnosť nadefinovať si nehnuteľnosť a viesť si záznam o histórii ubytovaných, prehľad príjmov a výdajov na daný mesiac, pridávať pripomienky a úlohy na splnenie i pridávať individuálne záznamy pre jednotlivé nehnuteľnosti. Aplikácie upozorní prenajímateľa na prípadne chýbajúce údaje a na udalosti, ktoré si prenajímateľ označil.

Prenajímateľ má možnosť pridávať k jednotlivým nehnuteľnostiam nájomníkov, ktorý si danú nehnuteľnosť prenajali. Aktuálne je však podporovaná možnosť priradiť jedného nájomníka do jednej nehnuteľnosti. Je to z dôvodu, že sa predpokladá, že jeden nájomník býva v jednej nehnuteľnosti a tiež zameraním aplikácie na užívateľov, u ktorých sa nepredpokladá veľké množstvo nehnuteľností. Možnosť prenájmu viacerých nehnuteľností jednou osobou je opäť možnosť budúceho rozšírenia aplikácie.

V súčasnosti je aplikácii navrhnutá tak, že je možné nadefinovať si nehnuteľnosť bez bližšej špecifikácie o aký konkrétny typ nehnuteľnosť sa jedná. Chcel som vytvoriť aplikáciu, ktorá podporovala základné funkcie a vlastnosti, ktoré majú všetky typy nehnuteľností spoločné. Myslím, že túto vlastnosť aplikácia splňuje. Medzi ďalšie rozšírenia, či úpravy, ktoré by mohli byť do aplikácie pridané patrí práve podpora rôznych typov nehnuteľnosti ako napr. byt, dom, prípadne samostatná izba v dome a pod..

Keďže sa predpokladá s určitými rozšíreniami aplikácie, je navrhnutá tak, aby ju šlo ďalej jednoducho rozširovať a pridávať ďalšie nové sekcie a funkcie, prípadne upravovať už existujúce. Z hľadiska funkčnosti a obsahu aplikácie si myslím, že všetky stanovené ciele, ktoré som si vytýčil v úvode pred jej programovaním aplikácia splňuje. Prenajímateľom poskytuje pohodlnú správu ich nehnuteľností a nájomníkov a nájomníkom poskytuje prehľadné informácie o ich prenájme s možnosťou úpravy údajov a komunikácie s prenajímateľom.

Zoznam použitej literatúry

[1] Dave Thomas, David Heinemeier Hansson (2006): Agile webdeveleopment with rails, 2nd edition. The Pragmatic Programmers LLC.

[2] Christian Heilmann (2006): Beginning JavaScript with DOM Scripting and Ajax From Novice to Professional. Apress(r)

[3] Chad Fowler (2006): Rails recipes. The Pragmatic Programmers LLC.

[4] Scott Raymond (2007): Ajax on Rails. O'Reilly Media, Inc.

[5] Steven Holzner (2007): Začíname programovat v Ruby On Rails. Computer Press a.s. 2007

[6] http://www.ruby-doc.org/core/

[7] http://api.rubyonrails.com/

[8] http://railsforum.com/

[9] http://railscasts.com/contest

[10] http://www.w3schools.com/

[11] http://www.pullmonkey.com/

[12] http://www.google.com